

PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the Application of

Taro TERA0

Application No.: New U.S. Patent Application

Filed: June 6, 2000

Docket No.: 106408

For: ONE-WAY FUNCTION GENERATION METHOD, ONE-WAY FUNCTION VALUE
GENERATION DEVICE, PROVING DEVICE, AUTHENTICATION METHOD, AND
AUTHENTICATION DEVICE



CLAIM FOR PRIORITY

Director of the U.S. Patent and Trademark Office
Washington, D.C. 20231

Sir:

The benefit of the filing date of the following prior foreign application filed in the following foreign country is hereby requested for the above-identified patent application and the priority provided in 35 U.S.C. §119 is hereby claimed:

Japanese Patent Application No. 11-244519 filed on August 31, 1999.

In support of this claim, a certified copy of said original foreign application:

 x is filed herewith.

 was filed on in Parent Application No. filed .

It is requested that the file of this application be marked to indicate that the requirements of 35 U.S.C. §119 have been fulfilled and that the Patent and Trademark Office kindly acknowledge receipt of this document.

Respectfully submitted,

James A. Oliff

Registration No. 27,075

JAO:TJP/crt

Thomas J. Pardini

Registration No. 30,411

Date: June 6, 2000

OLIFF & BERRIDGE, PLC
P.O. Box 19928
Alexandria, Virginia 22320
Telephone: (703) 836-6400

<p>DEPOSIT ACCOUNT USE AUTHORIZATION Please grant any extension necessary for entry; Charge any fee due to our Deposit Account No. 15-0461</p>

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

J-C833 U.S. PTO
09/588547
06/06/00

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日

Date of Application:

1999年 8月31日

出 願 番 号

Application Number:

平成11年特許願第244519号

出 願 人

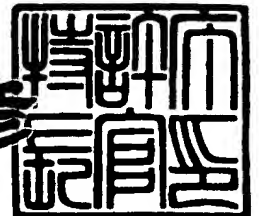
Applicant (s):

富士ゼロックス株式会社

2000年 4月21日

特許庁長官
Commissioner,
Patent Office

近 藤 隆 彦



出証番号 出証特2000-3028014

【書類名】 特許願

【整理番号】 FN99-00214

【提出日】 平成11年 8月31日

【あて先】 特許庁長官殿

【国際特許分類】 H04L 9/32

【発明の名称】 一方向性関数生成方法、一方向性関数値生成装置、証明装置、認証方法および認証装置

【請求項の数】 38

【発明者】

【住所又は居所】 神奈川県足柄上郡中井町境 4 3 0 グリーンテクなかい
富士ゼロックス株式会社内

【氏名】 寺尾 太郎

【特許出願人】

【識別番号】 000005496

【氏名又は名称】 富士ゼロックス株式会社

【電話番号】 0462-38-8516

【代理人】

【識別番号】 100086531

【弁理士】

【氏名又は名称】 澤田 俊夫

【電話番号】 03-5541-7577

【手数料の表示】

【予納台帳番号】 038818

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【ブルーフの要否】 要

【書類名】 明細書

【発明の名称】 一方向性関数生成方法，一方向性関数値生成装置，証明装置，
認証方法および認証装置

【特許請求の範囲】

【請求項 1】 一方向性関数 H と固有値 d に依存した一方向性関数 X を生成する生成方法であって、関数生成固有値 s を保持し、関数生成固有値 s と固有値 d より値生成固有値 u を作成し、値生成固有値 u とメッセージ M より一方向性関数 H を用いてメッセージ M の一方向性関数値 $X(M)$ を作成する一方向性関数生成方法。

【請求項 2】 請求項 1 に記載の一方向性関数生成方法であって、関数生成固有値 s と固有値 d より一方向性関数 G を用いて値生成固有値 u が計算されることを特徴とする一方向性関数生成方法。

【請求項 3】 請求項 1 に記載の一方向性関数生成方法であって、関数生成固有値 s と固有値 d より慣用暗号の暗号化関数 E を用いて値生成固有値 u が計算されることを特徴とする一方向性関数生成方法。

【請求項 4】 請求項 1 に記載の一方向性関数生成方法であって、値生成固有値 u とメッセージ M より一方向性関数 H と慣用暗号の暗号化関数 D を用いてメッセージ M の一方向性関数値 $X(M)$ が計算されることを特徴とする一方向性関数生成方法。

【請求項 5】 固有値 d に依存する一方向性関数 X を計算する一方向性関数値生成装置であって、固有値 d を入力する手段と、メッセージ M を入力する手段と、関数生成固有値 s を保持する手段と、関数生成固有値 s と固有値 d より値生成固有値 u を作成する手段と、値生成固有値 u とメッセージ M より一方向性関数 H を用いてメッセージ M の一方向性関数値 $X(M)$ を作成する手段とを有する一方向性関数値生成装置。

【請求項 6】 請求項 5 に記載の一方向性関数値生成装置であって、値生成固有値 u および一方向性関数値 $X(M)$ の計算過程を外部から観測することが困難なことを特徴とする一方向性関数値生成装置。

【請求項 7】 メッセージ M に依存する秘密鍵による処理を行う証明装置で

あって、メッセージ M を入力する手段と、値生成固有値 u を保持する手段と、値生成固有値 u とメッセージ M より一方向性関数 H を用いてメッセージ M の一方向性関数値 $X(M)$ を作成する手段と、秘密鍵 $X(M)$ による処理を行う手段とを有し、値生成固有値 u は、関数生成固有値 s と固有値 d より作成されることを特徴とする証明装置。

【請求項 8】 請求項 7 に記載の証明装置であって、値生成固有値 u および秘密鍵 $X(M)$ による処理の計算過程を外部から観測することが困難なことを特徴とする証明装置。

【請求項 9】 請求項 7 に記載の証明装置であって、スマートカードなどの携帯可能な小型演算装置として構成されることを特徴とする証明装置。

【請求項 10】 請求項 7 に記載の証明装置であって、CPU 内部のモジュールとして構成されることを特徴とする証明装置。

【請求項 11】 請求項 7 に記載の証明装置であって、秘密鍵による処理を行う手段が、チャレンジ c を入力する手段と、チャレンジ c と秘密鍵 $X(M)$ からレスポンス r を計算する手段と、レスポンス r を出力する手段からなることを特徴とする証明装置。

【請求項 12】 請求項 7 に記載の証明装置であって、秘密鍵による処理を行う手段が、チャレンジ c を入力する手段と、乱数 k を生成する手段と、乱数 k とチャレンジ c と秘密鍵 $X(M)$ からレスポンス r を計算する手段と、レスポンス r を出力する手段からなることを特徴とする証明装置。

【請求項 13】 請求項 7 に記載の証明装置であって、秘密鍵による処理を行う手段が、乱数 k を生成する手段と、乱数 k からコミットメント w を計算する手段と、コミットメント w を出力する手段と、チャレンジ c を入力する手段と、乱数 k とチャレンジ c と秘密鍵 $X(M)$ からレスポンス r を計算する手段と、レスポンス r を出力する手段からなることを特徴とする証明装置。

【請求項 14】 請求項 7 に記載の証明装置であって、秘密鍵による処理を行う手段が、乱数 k を生成する手段と、乱数 k からコミットメント w を計算する手段と、コミットメント w を出力する手段と、チャレンジ c を入力する手段と、乱数 k とコミットメント w とチャレンジ c と秘密鍵 $X(M)$ からレスポンス r を

計算する手段と、レスポンス r を出力する手段とを有することを特徴とする証明装置。

【請求項 15】 請求項 7 に記載の証明装置であって、秘密鍵による処理を行う手段が、有限体上の乗法群の乗法およびべき乗演算を行うことを特徴とする証明装置。

【請求項 16】 請求項 7 に記載の証明装置であって、秘密鍵による処理を行う手段が、有限体上の楕円曲線の加法およびスカラー倍演算を行うことを特徴とする証明装置。

【請求項 17】 請求項 7 に記載の証明装置であって、秘密鍵による処理を行う手段が、 n を因数分解が困難な合成数としてとして法 n のもとで乗算剰余およびべき乗剰余演算を行うことを特徴とする証明装置。

【請求項 18】 請求項 7 に記載の証明装置であって、メッセージ M は利用条件を含み、メッセージを入力する手段は、メッセージ M に含まれた利用条件が満たされない場合はメッセージの入力を拒絶することを特徴とする証明装置。

【請求項 19】 請求項 7 に記載の証明装置であって、メッセージ M は秘密鍵処理のパラメータを含み、秘密鍵による処理を行う手段は、メッセージ M に含まれた秘密鍵処理のパラメータに基づいて処理を行うことを特徴とする証明装置。

【請求項 20】 固有値 d に応じて証明器 T を発行する装置であって、固有値 d を入力する手段と、関数生成固有値 s を保持する手段と、関数生成固有値 s と固有値 d より値生成固有値 u を作成する手段と、値生成固有値 u を証明器 T に書き込む手段とを有し、証明器 T は、値生成固有値 u を保持しており、メッセージ M の入力に対して値生成固有値 u とメッセージ M より一方向性関数 H を用いてメッセージ M の一方向性関数値 $X(M)$ を作成して秘密鍵 $X(M)$ による処理を行うことを特徴とする証明器発行装置。

【請求項 21】 資格発行者が資格付与者にメッセージ M に対応づけて資格を発行し、資格検証者が資格付与者の資格を検証する認証方法であって、資格発行者は、関数生成固有値 s と資格付与者に対応する固有値 d より値生成固有値 u を作成し、値生成固有値 u とメッセージ M より一方向性関数 H を用いてメッセー

ジMの一方方向性関数値 $X(M)$ を計算し、秘密鍵 $X(M)$ と対をなす公開鍵 y を証明する証明証Cを資格付与者に発行し、資格付与者は、証明証Cを資格検証者に提示し、値生成固有値 u とメッセージMより一方方向性関数 H を用いてメッセージMの一方方向性関数値 $X(M)$ を計算し、秘密鍵 $X(M)$ による処理を行い、資格検証者は、証明証Cを検証し、また、資格付与者の秘密鍵 $X(M)$ による処理を証明証Cの証明する公開鍵 y で検証することによって認証を行う認証方法。

【請求項 2 2】 請求項 2 1 に記載の認証方法であって、資格発行者の発行する証明証Cには認証の種別を表す識別子 $a i d$ が記載され、資格検証者は、証明証Cに記載された認証識別子 $a i d$ が、行おうとする認証の種別に一致するときに限って、証明証Cの検証に成功することを特徴とする認証方法。

【請求項 2 3】 請求項 2 1 に記載の認証方法であって、資格発行者の発行する証明証Cには利用条件が記載され、資格検証者は、証明証Cに記載された利用条件が満たされるときに限って、証明証Cの検証に成功することを特徴とする認証方法。

【請求項 2 4】 固有値 d とメッセージMに応じて証明証Cを発行する証明証発行装置であって、固有値 d を入力する手段と、メッセージMを入力する手段と、関数生成固有値 s を保持する手段と、関数生成固有値 s と固有値 d より値生成固有値 u を作成する手段と、値生成固有値 u とメッセージMより一方方向性関数 H を用いてメッセージMの一方方向性関数値 $X(M)$ を作成する手段と、秘密鍵 $X(M)$ と対を成す公開鍵 y を作成する手段と、公開鍵 y を証明する証明証Cを発行する手段とからなる証明証発行装置。

【請求項 2 5】 メッセージMに応じて認証を行う認証装置であって、メッセージMを入力する手段と、値生成固有値 u を保持する手段と、値生成固有値 u とメッセージMより一方方向性関数 H を用いてメッセージMの一方方向性関数値 $X(M)$ を作成する手段と、秘密鍵 $X(M)$ による処理を行う手段と、秘密鍵 $X(M)$ と対をなす公開鍵 y を証明する証明証Cを保持する手段と、証明証Cを検証する手段と、公開鍵 y で秘密鍵による処理を検証する手段とからなり、値生成固有値 u は、関数生成固有値 s と固有値 d より作成されることを特徴とする認証装置。

【請求項 2 6】 資格発行者が資格付与者にメッセージ M に対応づけて資格を発行し、資格検証者が資格付与者の資格を検証する認証方法であって、資格発行者は、関数生成固有値 s と資格付与者に対応する固有値 d より値生成固有値 u を作成し、値生成固有値 u とメッセージ M より一方向性関数 H を用いてメッセージ M の一方向性関数値 $X(M)$ を計算し、秘密鍵 x と一方向性関数値 $X(M)$ とから定まるアクセスチケット t を資格付与者に発行し、資格付与者は、値生成固有値 u とメッセージ M より一方向性関数 H を用いてメッセージ M の一方向性関数値 $X(M)$ を計算し、秘密鍵 $X(M)$ による処理を行い、秘密鍵 $X(M)$ による処理をアクセスチケット t によって秘密鍵 x による処理に変換し、資格検証者は、資格付与者の秘密鍵 x による処理を秘密鍵 x と対をなす公開鍵 y で検証することによって認証を行う認証方法。

【請求項 2 7】 請求項 2 1 あるいは請求項 2 6 に記載の認証方法であって、メッセージ M には認証の種別を表す識別子 $a i d$ が含まれることを特徴とする認証方法。

【請求項 2 8】 固有値 d とメッセージ M に応じてアクセスチケット t を発行するアクセスチケット発行装置であって、固有値 d を入力する手段と、メッセージ M を入力する手段と、関数生成固有値 s を保持する手段と、関数生成固有値 s と固有値 d より値生成固有値 u を作成する手段と、値生成固有値 u とメッセージ M より一方向性関数 H を用いてメッセージ M の一方向性関数値 $X(M)$ を作成する手段と、秘密鍵 x と一方向性関数値 $X(M)$ よりアクセスチケット t を作成する手段と、アクセスチケット t を発行する手段とを有するアクセスチケット発行装置。

【請求項 2 9】 請求項 2 8 に記載のアクセスチケット発行装置であって、アクセスチケット t が、秘密鍵 x と一方向性関数値 $X(M)$ の差 $x - X(M)$ として計算されることを特徴とするアクセスチケット発行装置。

【請求項 3 0】 請求項 2 8 に記載のアクセスチケット発行装置であって、アクセスチケット t が、秘密鍵 x と一方向性関数値 $X(M)$ の商 $x / X(M)$ として計算されることを特徴とするアクセスチケット発行装置。

【請求項 3 1】 値生成固有値 u を (u_1, \dots, u_m) とし、一方向性関

数値 $X(M)$ を一方向性関数 H の値のビット連結 $H(u_1 | M) | \dots | H(u_m | M)$ から生成して所望のビット長とする請求項 28 記載のアクセスチケット生成装置。

【請求項 32】 値生成固有値 (u_1, \dots, u_m) を、関数生成固有値 s を (s_1, \dots, s_m) として一方向性関数 G を用いて $u_j = G(s_j | d)$ から求める請求項 31 記載のアクセスチケット生成装置。

【請求項 33】 メッセージ M に応じて認証を行う認証装置であって、メッセージ M を入力する手段と、値生成固有値 u を保持する手段と、値生成固有値 u とメッセージ M より一方向性関数 H を用いてメッセージ M の一方向性関数値 $X(M)$ を作成する手段と、秘密鍵 $X(M)$ による処理を行う手段と、秘密鍵 x と一方向性関数値 $X(M)$ とから定まるアクセスチケット t を保持する手段と、秘密鍵 $X(M)$ による処理をアクセスチケット t によって秘密鍵 x による処理に変換する手段と、秘密鍵 x と対をなす公開鍵 y を保持する手段と、公開鍵 y で秘密鍵による処理を検証する手段とを有し、値生成固有値 u は、関数生成固有値 s と固有値 d より作成されることを特徴とする認証装置。

【請求項 34】 請求項 33 に記載の認証装置であって、秘密鍵による処理を変換する手段が、チャレンジ c をアクセスチケット t によって更新する手段からなることを特徴とする認証装置。

【請求項 35】 請求項 33 に記載の認証装置であって、秘密鍵による処理を変換する手段が、レスポンス r をアクセスチケット t によって更新する手段からなることを特徴とする認証装置。

【請求項 36】 請求項 33 に記載の認証装置であって、秘密鍵による処理を変換する手段が、レスポンス r をアクセスチケット t とチャレンジ c によって更新する手段からなることを特徴とする認証装置。

【請求項 37】 請求項 33 に記載の認証装置であって、秘密鍵による処理を変換する手段が、チャレンジ c をコミットメント w によって更新する手段と、レスポンス r をアクセスチケット t とチャレンジ c によって更新する手段からなることを特徴とする認証装置。

【請求項 38】 請求項 33 に記載の認証装置であって、秘密鍵による処理

を変換する手段が、チャレンジ c をアクセスチケット t とコミットメント w によって更新する手段と、レスポンス r をコミットメント w によって更新する手段からなることを特徴とする認証装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、情報セキュリティ技術に関するものであり、特に、一方向性関数を生成する方法および一方向性関数値を生成する装置とそれらによる認証の方法および装置に関するものである。

【0002】

【従来の技術】

〔従来技術の概要〕

認証を行うための秘密鍵を、スマートカードなどの耐タンパー性を有する容器に封入して配布することは広く行われている。耐タンパー性容器に封入された秘密鍵を用いた次のような認証システムを構成することができる。

【0003】

説明の簡略化のため、秘密鍵の配布者をセンター、秘密鍵を格納する耐タンパー性容器をトークン、秘密鍵の受領者をユーザーと呼称する。

【0004】

センターは、秘密鍵 x を生成し、トークンに封入し、トークンをユーザーに配布する。トークンは、また、秘密鍵 x に基づいた処理を実行可能なように構成する。例えば、以下の様にすれば良い。

【0005】

センターは公開鍵暗号系を定める。例えば、 G を離散対数問題が困難な有限アーベル群（記法の簡略化のために加法的に記すが、もちろん、慣習的に乗法的に記される群であっても、以下の議論は単に記法を変更するのみで適用できる）、 p を素数、 F_p を p 元体、 $g : F_p \rightarrow G$ を体 F_p の加法群から有限群 G への自明でない準同型、 $y : F_p \rightarrow G$ を $y = g \cdot x$ （ここで x は体 F_p の加法群の x 倍自己準同型と同一視）で定義される準同型、 $\pi : G \rightarrow F_p$ を写像とする。このような有限

群 G として、ある有限体上の乗法群や楕円曲線を取ることができることは良く知られている。

【0006】

トークンは、入出力手段、乱数生成手段、写像 π g の計算手段と有限素体 F_p の算法の実行手段を持ち、入力されたチャレンジ c に対して、乱数 k を生成して、レスポンス $r = (r_0, r_1)$ を

【0007】

【数1】

$$r_0 = c - \pi(g(k))$$

$$r_1 = k - r_0 x$$

と計算して、出力する。

【0008】

一方、検証器は、入出力手段、乱数生成手段、準同型 g の計算手段、準同型 y の計算手段、有限群 G の算法の実行手段、有限素体 F_p の算法の実行手段とを持ち、チャレンジ c を乱数として生成して出力し、入力されたレスポンス r に対して、

【0009】

【数2】

$$c = r_0 + \pi(g(r_1) + y(r_0))$$

が成り立つことを検証する。

【0010】

以上の検証器とトークンを組み合わせることによって、検証器の送信したチャレンジ c に対して、秘密鍵 x によるチャレンジ c のNyberg-Rueppel署名をレスポンス r として返信することで認証を行うことができる。

【0011】

この手法は、例えば、アクセス制御の対象となるアプリケーションプログラムに検証器の機構を組み込み、そのアプリケーションプログラムの利用資格としてトークンを配布し、プログラムの実行時の要所要所で、上記の質疑応答による認証を行うことで、アクセス制御に応用することができる。アプリケーションプロ

グラム中の認証を行うコードは、攻撃者がこれらを削除する事を防ぐために、容易に発見されないようにしなければならない。

【0012】

上に説明した手法を直截に適用すると、例えば、同様な手法でアクセス制御された独立なアプリケーションプログラムを複数利用する場合、必要な認証の種類が増加し、認証の種類の数だけのトークンが必要になるが、以下に説明するように、一つのトークンのみで複数の認証に対応することもできる。

【0013】

この例では、センターは各トークン毎に異なる秘密鍵 x を封入して、ユーザーに配布する。また、各アプリケーションプログラムには認証識別子 $a i d$ が割り当てられている。アプリケーションプログラムの提供者がユーザーに認証識別子 $a i d$ に対応したアプリケーションプログラムの利用権を付与する場合、認証識別子 $a i d$ とユーザーの所持するトークンの公開鍵 y とを含み、アプリケーションプログラムの提供者の署名が施された証明証 C をユーザーに発行する。

【0014】

アプリケーションプログラムは、プログラムの実行時、ユーザーの利用資格を認証するタイミングで、証明証 C を読み込んで署名を検証し、記載された認証識別子 $a i d$ が自身の認証識別子に相等しいことを確認し、記載されたトークンの公開鍵 y に基づいた認証を前述のように実行する。

【0015】

アクセス資格認証のケーパビリティとしてトークンに格納された秘密鍵に対応する公開鍵を含む証明証を用いる方法は、秘密鍵をトークンに封入して配布するセンターが信用できる第三者と仮定すると、複数の互いに利害関係を持たないアプリケーションプログラムの提供者が共通に利用することができ、またユーザーも単一のトークンを保持するだけで済むので優れている。また、例えば、トークンのモジュールを $C P U$ に予め内蔵する構成をとれば、センターがユーザーにトークンを配布する手間をなくし、ユーザーもトークンを意識することなく利用することができる。

【0016】

一方、このような方法は、トークンの公開鍵が大域的なユーザーの識別子となり、ケーパビリティが大規模に収集された場合、トークンの公開鍵で名寄せが行われるなど、ユーザーのプライバシーの観点からは望ましくない。

【0 0 1 7】

一方、耐タンパー性容器に秘密ハッシュ関数を封入して配布することによって、上の問題点を解消する次のような認証システムを構成することができる。

【0 0 1 8】

この場合、センターは、秘密ハッシュ関数 X を生成し、トークンに封入し、トークンをユーザーに配布する。ここで、秘密ハッシュ関数 X は各トークン毎に異なるものとする。

【0 0 1 9】

アプリケーションプログラムの提供者がユーザーに利用資格を表すケーパビリティを前述したように証明証 C の形で付与する場合、例えば、認証識別子 $a i d$ の秘密ハッシュ関数 X によるハッシュ値 $X(a i d)$ に対応するトークンの公開鍵 $y = g(X(a i d))$ を証明証 C に含ませる。

【0 0 2 0】

今回は認証識別子 $a i d$ 毎に対応する公開鍵 y が異なったものになるので、ケーパビリティを収集しても、トークンの公開鍵による名寄せは不可能になる。

【0 0 2 1】

利用資格の認証を行う場合には、トークンに認証を行う認証識別子 $a i d$ が入力され、これに対応する秘密鍵 $x = X(a i d)$ が生成されるステップが増えるのみで後は同様である。

【0 0 2 2】

また、トークンがトークン毎に異なる秘密ハッシュ関数 X を持つものとして、特開平 1 0 - 2 4 7 9 0 5 号公報のアクセス資格認証装置では、以下に説明するような、上に説明したものとは異なるケーパビリティが提案されている。

【0 0 2 3】

ここでは、 $y = g(x)$ を満たす公開鍵 y と秘密鍵 x の対は、各ユーザーが所持するトークン内での秘密ハッシュ値 $X(a i d)$ とは独立に認証識別子 $a i d$

にのみ依存するように定められ（例えば，認証識別子 $a i d$ を公開鍵 y 自身として良い），ユーザーに対して発行するケーパビリティを $t = x - X(a i d)$ とする。このように秘密鍵 x とトークン内での秘密ハッシュ値 $X(a i d)$ から計算される量であるケーパビリティをアクセスチケットと呼ぶことにする。

【0024】

検証器は，自分の識別子 $a i d$ に対応する公開鍵 y を保持し，前述の質疑応答による認証を行う。

【0025】

また，ユーザーは，トークンに $a i d$ を入力して，トークン内で秘密ハッシュ値 $X(a i d)$ を生成し，チャレンジ c に対して，秘密鍵 $X(a i d)$ を用いてレスポンス $r = (r_0, r_1)$ を取得する。

【0026】

ユーザーは，さらに，トークンの出力したレスポンス r をアクセスチケット t によって，

【0027】

【数3】

$$r_1 \leftarrow r_1 + r_0 t$$

として更新した後に，レスポンス r を検証器に返信する。

【0028】

更新されたレスポンス r は，チャレンジ c に対する秘密鍵 x による $N y b e r g - R u e p p e l$ 署名であることは容易に確認できる。

【0029】

アクセスチケットによる認証方式では，アプリケーションプログラムはケーパビリティを読み込んだり，ケーパビリティの真正性を検証することを省略でき，検証器側（つまり，アプリケーションプログラムのアクセス制御コード）の構造はさらに簡易化され効率が良い。

【0030】

また，アクセスチケットによる認証方式は，トークン内で生成される秘密ハッシュ値とは独立な秘密鍵による処理を，秘密鍵を開示することなく実行可能とす

るので、単に、質疑応答型の認証に限らず、秘密鍵による処理（例えば、公開鍵暗号における暗号文の復号やメッセージへの署名生成）を行う機能を安全に委託することを可能にするという著しい特徴を備えている。

【0031】

以上で述べてきた証明証やアクセスチケットによるアクセス制御の方式は、もちろん、アプリケーションプログラムの実行制御のみならず、例えば、ファイルへのアクセス制御やhttpサーバーへのアクセス制御等にも適用することが可能である。

【0032】

また、トークンを可搬性を持ったスマートカードのような形態で提供し、ケーパビリティをトークンに格納することによって、これらのデジタル情報であるケーパビリティで通常のチケットやカードを模倣することも可能である。

【0033】

このように、トークンに秘密鍵の代わりに秘密ハッシュ関数を封入することにより、多彩な認証方式が利用できるようになる。

【0034】

〔従来技術の問題点〕

ハッシュ関数は、通常、暗号プロトコルを構成するプリミティブとしては公開される固定的な関数として利用されるケースが多く、暗号用のICチップにおいても、SHA-1 (SHA: Secure Hash Algorithm) や MD5 (MD: Message Digest) といった標準的なハッシュ関数のみを備えている場合が多いので、トークンの実装コストを考慮すると、これらの標準的なハッシュ関数を流用して、トークン毎に異なる秘密ハッシュ関数を実現する方法を吟味する必要がある。

【0035】

一つの簡単な方法は、トークンは標準的なハッシュ関数Hとトークン毎に異なる秘密固有値uを保持し、秘密ハッシュ関数Xを $X(M) = H(u \parallel M)$ として実現することである。

【0036】

この場合、センターはトークン毎に乱数として u を生成し、トークンの識別子 $t i d$ とトークン秘密固有値 u とのタプル $(t i d, u)$ をデータベースに保持して置くことが考えられるが、トークンの個数が増加した場合、データ量が膨大となり、また、このデータベースのエントリーは、上記認証システムを構成する上で厳重に守秘されることが必須であるので管理が困難である。

【0037】

例えば、証明証型のケーパビリティを用いる場合、一つのトークンの秘密ハッシュ関数 X が漏洩し、ケーパビリティの発行者がその漏洩の事実気付かない限り、以後、このトークンと認証識別子 $a i d$ に対応する証明証 C を発行させるたびに、証明証 C とそれに対応する秘密鍵 $X(M)$ を知る者はトークンを用いることなく検証にパスできるようになる。ただし、この場合は秘密ハッシュ値 $X(M)$ から漏洩者を追跡することができる。

【0038】

アクセスチケット型のケーパビリティを用いる場合、一つのトークンの秘密ハッシュ関数 X が漏洩し、ケーパビリティの発行者がその漏洩の事実気付かない限り、以後、このトークンと認証識別子 $a i d$ に対応するアクセスチケット t を発行させるたびに、 $x = t + X(M)$ として認証識別子 $a i d$ に対応した秘密鍵 x が組織的に暴露され深刻な問題となる。しかも、この場合は、暴露された秘密鍵 x 自身からは漏洩者を追跡できない。

【0039】

センターが厳重に秘密保持するデータ量を削減するために、センター秘密固有値 s を導入し、トークン毎の秘密固有値を持たない構成を考える。ここでは、秘密ハッシュ関数 X は $X(M) = H(s \parallel t i d \parallel M)$ として実現される。

【0040】

このような構成では、センターは唯一の秘密情報 s のみを安全に管理すれば、各トークン毎の秘密固有情報にあたるものを保持しておく必要が無いが、トークンには大域的な秘密情報 s が封入されることになり、万一、一つのトークンで耐タンパー性の仮定が破られた場合、トークン識別子 $t i d$ を入手可能なデータと考えると全てのトークンの秘密ハッシュ関数が一斉に暴露されてしまい危険が大

きい。

【0041】

また、今まで述べてきた認証方式は、例として挙げた離散対数問題が困難な有限群 G 上のNyberg-Rueppel署名だけでなく、様々な公開鍵暗号系の上で構成できる。

【0042】

まず、離散対数問題の困難性に立脚したDiffie-Hellman鍵共有、DSA署名(Digital Signature Algorithm)、Schnorr認証等の公開鍵暗号系には全て同様に適用できる。

【0043】

また、零化域決定問題の困難性に立脚したRSA, Guillou-Quisquater認証等の公開鍵暗号系にも適用できる。

【0044】

この場合、 λ を有限アーベル群 G の全ての元 γ を零化(つまり $\lambda \gamma = 0$)する最小の非零整数とし、有限群 G において λ の決定が困難とする。このような有限群 G として、有理整数環 \mathbb{Z} のRSA法数 n による剰余類環の乗法群 $(\mathbb{Z}/n\mathbb{Z})^*$ を取ることができることは良く知られている。

【0045】

例えば、公開鍵暗号系として有限群 G 上のGuillou-Quisquater署名を採用し、秘密ハッシュ値 $X(aid)$ を秘密鍵として公開鍵 $y = pX(aid)$ を証明する証明証 C をケーパビリティとして利用することもできるし、また、秘密鍵 x と公開鍵 $y = px$ に対応するアクセスチケット $t = x - X(aid)$ をケーパビリティとして利用することもできる。

【0046】

また、公開鍵暗号系として有限群 G 上のRSAを選び、公開鍵 y と秘密鍵 $x = y^{-1}$ に対応するアクセスチケット $t = x - X(aid)$ をケーパビリティとして利用することもできる。

【0047】

これらの諸例では、秘密ハッシュ値 $X(aid)$ は公開鍵暗号系の秘密値が属

する有限代数系の元とみなされている。秘密値が属する代数系は、Nyberg-Rueppel 署名等の離散対数問題系の例では、有限素体 F_p であり、Guillou-Quisquater 署名や RSA 等の零化域決定問題系の例では、それぞれ、有限群 G 自体とその忠実な作用環 $Z/\lambda Z$ である。

【0048】

離散対数問題系の例では、現在、推奨されている秘密鍵サイズ（160ビット程度）は、通常使用されているハッシュ関数の値のサイズと同程度であるが、零化域決定問題系の例では、推奨されている秘密鍵サイズ（1024ビット程度）は、通常使用されるハッシュ関数の値のサイズよりも大きい。

【0049】

【発明が解決しようとする課題】

以上に説明してきたように、本発明は、センターの管理コストが低く、また、秘密ハッシュ関数を封入されたトークンが万一開封された場合も、秘密情報の暴露が小規模に押さえられ、また、標準のハッシュ関数を流用することで、トークンの開発コストが低く、また、利用する公開鍵暗号の秘密鍵の鍵サイズと同等のサイズの値を有する、秘密ハッシュ関数を配布する方式を提案することを目的としている。

【0050】

このような秘密ハッシュ関数の配布のためには、標準のハッシュ関数があるパラメータによって変形したハッシュ関数の族を組織的に生成するハッシュ関数生成方式であって、しかも、その方式で生成されたハッシュ関数の計算において、ハッシュ関数の生成に用いたパラメータ自身を利用しないものがあれば良い。

【0051】

【課題を解決するための手段】

本発明のハッシュ関数生成スキームは、既知のハッシュ関数 H と固有値 d に依存したハッシュ関数 X を生成するにあたって、センターがハッシュ関数生成固有値 s を保持し、ユーザーには、ハッシュ関数生成固有値 s と固有値 d より計算されたハッシュ値生成固有値 u を封入したハッシュ値生成装置を配布するものであり、ハッシュ値生成固有値 u とメッセージ M よりハッシュ関数 H を用いてメッセ

ージMのハッシュ値X (M) を生成するものである。

【0052】

本発明においては、センターのハッシュ関数生成固有値とトークン（ユーザ）の固有値からトークンごとに異なるハッシュ値生成固有値uを生成するのでトークンごとに秘密固有値を保持管理する必要がない。また、トークンに保持されているハッシュ値生成固有値uが漏洩しても組織的な秘密情報（センターのハッシュ関数生成固有値s）が暴露されるおそれがない。

【0053】

本発明は特許請求の範囲に記載のとおり一方向性関数生成方法、一方向性関数値生成装置、証明装置、証明器発行装置、認証方法、認証装置、アクセスチケット発行装置として実現することができる。また本発明の少なくとも一部をソフトウェア製品として実現することもできる。

【0054】

【発明の実施の形態】

まず最初に、以後の説明で共通に使われる記号を説明する。

【0055】

Z は有理整数環、 p は素数、 F_p は p 元体、 $\{0, 1\}^*$ はビット列のモノイド（算法はビット列の連結であり $|$ で表す）である。

【0056】

G は有限アーベル群、 $\pi : G \rightarrow F_p$ と $\varepsilon : G \rightarrow \{0, 1\}^*$ と $\eta : \{0, 1\}^* \rightarrow F_p$ は写像とする。

【0057】

記述を簡略化するために有限群 G を加法的に記す。また、 r を有限群 G の元、 v を有理整数としたとき、 r の v 倍を必ず左作用の形で $v \cdot r$ と書くことにする。有限群 G が乗法的に記されている場合は 0 を 1 に、 $r + r'$ を $r r'$ に、 $-$ を $/$ に、 $v \cdot r$ を

【0058】

【数4】

$$r^v$$

に置き換えれば良い。

【0059】

また、 g は位数が p の有限群 G の元、 λ は有限群 G の零化域

【0060】

【数5】

$$\text{Ann } G = \{v \in \mathbb{Z} ; (\forall \gamma \in G) v \cdot \gamma = 0\}$$

の生成元とし、 Λ は有理整数環 \mathbb{Z} の法 λ による剰余類環 $\mathbb{Z} / \lambda \mathbb{Z}$ とする。有限環 Λ は有限群 G の忠実な作用環（つまり Λ の元 α による G の元 γ のスカラー倍がきちんと定義され、 Λ の元 α が G の任意の元 γ について $\alpha \cdot \gamma = 0$ ならば $\alpha = 0$ ）である。

【0061】

有限群 G の与えられた元 $y = x \cdot g$ から p 元体 F_p の元 x を求めることを底 g についての離散対数問題、有理整数 λ を求めることを零化域決定問題とよぶことにしよう。以後の実施例で現れる有限群 G は、離散対数問題が困難であるか、零化域決定問題が困難であるかのいずれかである。

【0062】

離散対数問題が困難な有限群 G としては、 q をある素数の冪として有限体 F_q 上の乘法群 GL_1 ないし楕円曲線 E が良く知られている。また、零化域決定問題が困難な有限群 G としては、有理整数環 \mathbb{Z} の合成数 n による剰余類環の乘法群が良く知られている。

【0063】

写像 $\pi : G \rightarrow F_p$ として、ハッシュ関数、法 p による剰余、また、 $q < p$ として有限群 G が有限体 F_q 上の楕円曲線の場合は座標関数（つまり、 $P = (x_p, y_p)$ に対して、 $\pi(P) = x_p$ とする）等がある。

【0064】

写像 $\varepsilon : G \rightarrow \{0, 1\}^*$ として、有限群 G の元の自然な符号化（つまり、ビット列としての表現）を対応させる関数やハッシュ関数等がある。

【0065】

写像 $\eta : \{0, 1\}^* \rightarrow F_p$ として、ハッシュ関数等がある。

【0066】

[実施例 1 : ハッシュ値生成装置]

本実施例では、固有値 d とメッセージ M を入力すると固有値に依存したハッシュ関数 X によるメッセージ M のハッシュ値 $X(M)$ を出力するハッシュ値生成装置を図 1 に基づいて説明する。図 1 はハッシュ値生成装置の構成図である。

【0067】

固有値入力部 1 は、ハッシュ関数 X の生成に要するパラメータである固有値 d が入力される。メッセージ入力部 2 は、ハッシュ値を求めたいメッセージ M が入力される。関数生成固有値記憶部 3 は、値生成固有値の生成に要するパラメータである関数生成固有値 s を保持する。値生成固有値算出部 4 は、関数生成固有値記憶部 3 に保持された関数生成固有値 s と固有値入力部 1 に入力された固有値 d より値生成固有値 u を生成する。ハッシュ値算出部 5 は、値生成固有値算出部 4 が生成した値生成固有値 u とメッセージ入力部 2 に入力されたメッセージ M よりハッシュ関数 H を用いてハッシュ値 $X(M)$ を生成する。ハッシュ値出力部 6 は、ハッシュ値算出部 5 が生成したハッシュ値 $X(M)$ を出力する。

【0068】

ハッシュ関数 H として、例えば、RSA Data Security, Inc. の MD5 (ハッシュ値長は 128 ビット) や米国 National Institute of Standards and Technology の FIPS 180-1 で定義された SHA-1 (ハッシュ値長は 160 ビット) を用いて良い。

【0069】

以下、値生成固有値 u およびハッシュ値 $X(M)$ の具体的な生成方法を、様々なバリエーションを交えて説明しよう。

【0070】

値生成固有値算出部 4 は、値生成固有値 u を、例えば、ハッシュ関数 G を用いて、 $u = G(s \parallel d)$ として計算しても良い。ここでも、ハッシュ関数 G として MD5 や SHA-1 を取ることができる。また、ハッシュ関数 G がハッシュ関数 H と同一でも良い。

【0071】

また、例えば、慣用暗号の暗号化関数 E を用いて、 $u = E(d, s)$ として計算しても良い。ここで、固有値 d を暗号化鍵とすると、値生成固有値 u のサイズは関数生成固有値 s のサイズと同じになる。

【0072】

ハッシュ値算出部 5 は、ハッシュ値 $X(M)$ を、例えば、 $X(M) = H(u \parallel M)$ として計算しても良い。このようにしてハッシュ値 $X(M)$ を計算すると、ハッシュ値 $X(M)$ の長さはハッシュ関数 H のハッシュ値長と同じになる。

【0073】

ハッシュ値 $X(M)$ の長さをハッシュ関数 H のハッシュ値長より大きくする方法を説明しよう。

【0074】

値生成固有値算出部 4 は、固有値 d と関数生成固有値 $s = (s_1, \dots, s_m)$ に対して、値生成固有値 $u = (u_1, \dots, u_m)$ を、例えば、ハッシュ関数 G を用いて、 $u_i = G(s_i \parallel d)$ と計算する。あるいは、固有値 d を $d = (d_1, \dots, d_m)$ として、 $u_i = G(s_i \parallel d_i)$ として計算しても良い。

【0075】

ハッシュ値算出部 5 は、ハッシュ値 $X(M)$ を $X(M) = H(u_1 \parallel M) \parallel \dots \parallel H(u_m \parallel M)$ として計算する。このようにするとハッシュ値 $X(M)$ の長さはハッシュ関数 H のハッシュ値長の m 倍となり、例えば、ハッシュ関数 H として 128 ビットのハッシュ値を持つ MD5 を取り、 $m = 8$ とするとハッシュ値 $X(M)$ は 1024 ビットとなる。あるいは、ハッシュ関数 H として 160 ビットのハッシュ値を持つ SHA-1 を取り、 $m = 7$ とし、ハッシュ値 $X(M)$ を $H(u_1 \parallel M) \parallel \dots \parallel H(u_m \parallel M)$ の上位 1024 ビットとしても、1024 ビットのハッシュ値 $X(M)$ を得ることができる。

【0076】

また、例えば、値生成固有値 u とメッセージ M よりハッシュ関数 H と慣用暗号の暗号化関数 E を用いてハッシュ値 $X(M)$ を $X(M) = E(H(u \parallel M), u)$ として計算する。ここでは、 E の最初の変数が暗号化の鍵とすると、ハッシュ

値 X (M) の長さは値生成固有値 u の長さと等しくなる。値生成固有値 u の長さが大きくなるようにしておけば、このようにしても長いハッシュ値を得られる。

【 0 0 7 7 】

[実施例 2 : 証明装置 (確定型 2 方向認証)]

本実施例では、メッセージ M を入力するとメッセージ M に依存した秘密鍵 X (M) による処理を行う証明装置を図 2 と図 3 に基づいて説明する。図 2 は証明装置の構成図である。本実施例の証明装置は、ハッシュ値生成装置 (実施例 1) で用いたものと等価なメッセージ入力部 2 とハッシュ値算出部 5 を持つ。図 3 は秘密鍵処理部 8 の詳細な構成図である。

【 0 0 7 8 】

値生成固有値記憶部 7 は、ハッシュ値 X (M) の生成に要するパラメータである値生成固有値 u を保持する。ハッシュ値算出部 5 は、値生成固有値記憶部 7 に保持された値生成固有値 u とメッセージ入力部 2 に入力されたメッセージ M よりハッシュ関数 H を用いてハッシュ値 X (M) を生成する。秘密鍵処理部 8 は、ハッシュ値算出部 5 の生成したハッシュ値 X (M) を秘密鍵として処理を行う。

【 0 0 7 9 】

この証明装置は、例えば、スマートカード等の携帯性を備えたデバイスとして実現しても良い。このようにすると証明装置を常時持ち運び、いろいろな局面で利用することができる。

【 0 0 8 0 】

また、この証明装置は、例えば、CPU の内部モジュールとして実現してもよい。証明装置を主に計算機の利用時のアクセス資格証明に用いるのであれば、CPU に予め内蔵された証明装置は、計算機とは別途に入手する必要がなく、また、この証明装置を用いたアクセス資格認証によって、コンテンツを保護したいと考えるコンテンツ提供者側から見ても、証明装置の頒布の手間が省けるので好都合である。

【 0 0 8 1 】

メッセージ入力部 2 は、入力されるメッセージ M に基づいた処理を行ってもよい。

【 0 0 8 2 】

例えば、メッセージMが有効期限情報等の利用条件を含み、利用条件が満たされない場合は、メッセージの入力を拒絶するように構成してもよい。このように構成することで、秘密鍵による処理の実行に有効期限情報等の利用条件を対応づけることが可能となる。

【 0 0 8 3 】

また、メッセージMが秘密鍵処理のパラメータ G 、 p 、 π 、 ε 、 η 、 g 等を含んでいてもよい。このように構成することで、メッセージMに対応づけて秘密鍵処理のパラメータを変更することが可能となる。

【 0 0 8 4 】

また、メッセージMが秘密鍵処理のアルゴリズムの識別子を含んでいてもよい。このように構成することで、メッセージMに対応づけて秘密鍵処理のアルゴリズムを変更することが可能となる。

【 0 0 8 5 】

秘密鍵処理部 8 は、図 3 に示すように、チャレンジ入力部 9 とレスポンス生成部 1 0 とレスポンス出力部 1 1 より構成される。

【 0 0 8 6 】

チャレンジ入力部 9 は、認証のための質疑情報であるチャレンジ c が入力される。レスポンス生成部 1 0 は、チャレンジ入力部 9 に入力されたチャレンジ c とハッシュ値算出部 6 が生成したハッシュ値 $X(M)$ よりレスポンス r を生成する。レスポンス出力部 1 1 は、レスポンス生成部 1 0 が生成したレスポンス r を出力する。

【 0 0 8 7 】

レスポンス r の生成方法をD i f f i e - H e l l m a n鍵共有とR S Aを例にとって説明しよう。

【 0 0 8 8 】

有限群 G は、D i f f i e - H e l l m a n鍵共有の例では離散対数問題が困難とし、R S Aの例では零化域決定問題が困難とする。ハッシュ値算出部 6 が生成するハッシュ値 $X(M)$ は、D i f f i e - H e l l m a n鍵共有の例では p

元体 F_p の元であり、RSA の例では有限環 Λ の元である。チャレンジ入力部 9 に入力されるチャレンジ c は有限群 G の元である。

【0089】

[Diffie-Hellman 鍵共有の場合]

レスポンス r は

【0090】

【数 6】

$$r = X(M) \cdot c$$

として生成される。

【0091】

[RSA の場合]

レスポンス r は

【0092】

【数 7】

$$r = X(M) \cdot c$$

として生成される。

【0093】

[実施例 3：証明装置（ランダム型 2 方向認証）]

本実施例では、実施例 2 と異なる秘密鍵処理部 8 を持つ証明装置の例を図 4 に基づいて説明している。図 4 は秘密鍵処理部 8 の詳細な構成図である。

【0094】

秘密鍵処理部 8 は、乱数生成部 12 とチャレンジ入力部 9 とレスポンス生成部 10 とレスポンス出力部 11 より構成される。

【0095】

乱数生成部 12 は、乱数 k を生成する。レスポンス生成部 10 は、乱数生成部 12 の生成した乱数 k とチャレンジ入力部 9 に入力されたチャレンジ c とハッシュ値算出部 6 が生成したハッシュ値 $X(M)$ よりレスポンス r を生成する。

【0096】

以下、レスポンス r の具体的な生成方法を、様々なバリエーションを交えて説

明しよう。

【0097】

有限群Gは、DSA署名、ElGamal署名の変種、Nyberg-Rueppel署名、Schnorr署名の例では離散対数問題が困難とし、メッセージ復元型Guillou-Quisquater署名、Guillou-Quisquater署名の例では零化域決定問題が困難とする。

【0098】

ハッシュ値算出部6が生成するハッシュ値X(M)と乱数生成部12が生成する乱数kは、DSA署名、ElGamal署名の変種、Nyberg-Rueppel署名、Schnorr署名の例ではp元体 F_p の元であり、メッセージ復元型Guillou-Quisquater署名、Guillou-Quisquater署名の例では有限群Gの元である。

【0099】

チャレンジ入力部9に入力されるチャレンジcは、DSA署名、ElGamal署名の変種、Nyberg-Rueppel署名、メッセージ復元型Guillou-Quisquater署名の例では、p元体 F_p の元であり、Schnorr署名、Guillou-Quisquater署名の例では任意のビット列である。

【0100】

[DSA署名の場合]

レスポンスrは

【0101】

【数8】

$$r = (r_0, r_1)$$

$$r_0 = \pi(k, g)$$

$$r_1 = (c + r_0 X(M)) / k$$

として生成される。

【0102】

[変形ElGamal署名1の場合]

レスポンス r は

【0 1 0 3】

【数 9】

$$r = (r_0, r_1)$$

$$r_0 = \pi(k, g)$$

$$r_1 = ck - r_0 X(M)$$

として生成される。

【0 1 0 4】

【変形 E 1 G a m a 1 署名 2 の場合】

レスポンス r は

【0 1 0 5】

【数 1 0】

$$r = (r_0, r_1)$$

$$r_0 = \pi(k, g)$$

$$r_1 = r_0 k - c X(M)$$

として生成される。

【0 1 0 6】

【変形 E 1 G a m a 1 署名 3 の場合】

レスポンス r は

【0 1 0 7】

【数 1 1】

$$r = (r_0, r_1)$$

$$r_0 = \pi(k, g)$$

$$r_1 = (r_0 + c X(M)) / k$$

として生成される。

【0 1 0 8】

【変形 E 1 G a m a 1 署名 4 の場合】

レスポンス r は

【0 1 0 9】

【数 1 2】

$$\begin{aligned} r &= (r_0, r_1) \\ r_0 &= \pi(k, g) \\ r_1 &= (ck - r_0) / X(M) \end{aligned}$$

として生成される。

【0 1 1 0】

[変形 E 1 G a m a 1 署名 5 の場合]

レスポンス r は

【0 1 1 1】

【数 1 3】

$$\begin{aligned} r &= (r_0, r_1) \\ r_0 &= \pi(k, g) \\ r_1 &= (r_0 k - c) / X(M) \end{aligned}$$

として生成される。

【0 1 1 2】

[N y b e r g - R u e p p e l 署名の場合]

レスポンス r は

【0 1 1 3】

【数 1 4】

$$\begin{aligned} r &= (r_0, r_1) \\ r_0 &= c - \pi(k, g) \\ r_1 &= k - r_0 X(M) \end{aligned}$$

として生成される。

【0 1 1 4】

[S c h n o r r 署名の場合]

レスポンス r は

【0 1 1 5】

【数 1 5】

$$r = (r_0, r_1)$$

$$r_0 = \eta(c \parallel \varepsilon(k, g))$$

$$r_1 = k + r_0 X(M)$$

として生成される。

【0116】

[メッセージ復元型 Guillou-Quisquater 署名の場合]

レスポンス r は

【0117】

【数16】

$$r = (r_0, r_1)$$

$$r_0 = c - \pi(p, k)$$

$$r_1 = k - r_0 \cdot X(M)$$

として生成される。

【0118】

[Guillou-Quisquater 署名の場合]

レスポンス r は

【0119】

【数17】

$$r = (r_0, r_1)$$

$$r_0 = \eta(c \parallel \varepsilon(p, k))$$

$$r_1 = k + r_0 \cdot X(M)$$

として生成される。

【0120】

[実施例4：証明装置（3方向認証）]

本実施例では、実施例2ないし3と異なる秘密鍵処理部8を持つ証明装置の例を図5に基づいて説明している。図5は秘密鍵処理部8の詳細な構成図である。

【0121】

秘密鍵処理部8は、乱数生成部12とコミットメント生成部13とコミットメント出力部13とチャレンジ入力部9とレスポンス生成部10とレスポンス出力部11より構成される。

【0122】

乱数生成部12は、乱数 k を生成する。コミットメント生成部13は、乱数生成部12が生成した乱数 k よりコミットメント w を生成する。コミットメント出力部14は、コミットメント生成部13が生成したコミットメント w を出力する。レスポンス生成部10は、乱数生成部12の生成した乱数 k とチャレンジ入力部9に入力されたチャレンジ c とハッシュ値算出部6が生成したハッシュ値 $X(M)$ よりレスポンス r を生成する。

【0123】

乱数生成部12で生成された乱数 k は、レスポンス生成部10でレスポンス r の生成に利用された直後に破棄するなどして、同一の乱数 k が異なるレスポンス r の生成に重複して利用されないように注意しなければならない。

【0124】

以下、レスポンス r の具体的な生成方法を、Schnorr認証、Guillou-Quisquater認証、Fiat-Shamir認証を例にとって説明しよう。

【0125】

有限群 G は、Schnorr認証の例では離散対数問題が困難とし、Guillou-Quisquater認証、Fiat-Shamir認証の例では零化域決定問題が困難であるとする。

【0126】

ハッシュ値算出部6が生成するハッシュ値 $X(M)$ と乱数生成部12が生成する乱数 k は、Schnorr認証の例では p 元体 F_p の元であり、Guillou-Quisquater認証、Fiat-Shamir認証の例では有限群 G の元である。

【0127】

チャレンジ入力部9に入力されるチャレンジ c は、Schnorr認証、Guillou-Quisquater認証の例では p 元体 F_p の元であり、Fiat-Shamir認証の例では固定長のビット列である。

【0128】

[Schnorr 認証 1 の場合]

コミットメント w は

【0 1 2 9】

【数 1 8】

$$w = \pi(k, g)$$

として生成され, レスポンス r は

【0 1 3 0】

【数 1 9】

$$r = k - cX(M)$$

として生成される。

【0 1 3 1】

[Schnorr 認証 2 の場合]

コミットメント w は

【0 1 3 2】

【数 2 0】

$$w = k, g$$

として生成され, レスポンス r は

【0 1 3 3】

【数 2 1】

$$r = k - cX(M)$$

として生成される。

【0 1 3 4】

[Guillou-Quisquater 認証 1 の場合]

コミットメント w は

【0 1 3 5】

【数 2 2】

$$w = \pi(p, k)$$

として生成され, レスポンス r は

【0 1 3 6】

【数 23】

$$r = k - c \cdot X(M)$$

として生成される。

【0137】

[Guillou-Quisquater 認証 2 の場合]

コミットメント w は

【0138】

【数 24】

$$w = p \cdot k$$

として生成され、レスポンス r は

【0139】

【数 25】

$$r = k - c \cdot X(M)$$

として生成される。

【0140】

[Fiat-Shamir 認証 1 の場合]

コミットメント w は

【0141】

【数 26】

$$w = \pi(2, k)$$

として生成され、レスポンス r は

【0142】

【数 27】

$$r = k - \sum c_i \cdot X(M)_i$$

として生成される。ただし、

【0143】

【数 28】

$$c = (c_1, \dots, c_n)$$

$$X(M) = (X(M)_1, \dots, X(M)_n)$$

である。

【0 1 4 4】

[F i a t - S h a m i r 認証 2 の場合]

コミットメント w は

【0 1 4 5】

【数 2 9】

$$w = 2. k$$

として生成され、レスポンス r は

【0 1 4 6】

【数 3 0】

$$r = k - \sum c_i \cdot X(M)_i$$

として生成される。ただし、

【0 1 4 7】

【数 3 1】

$$c = (c_1, \dots, c_n)$$

$$X(M) = (X(M)_1, \dots, X(M)_n)$$

である。

【0 1 4 8】

[実施例 5 : 証明装置 (擬 3 方向認証)]

本実施例では、実施例 2 ないし 4 と異なる秘密鍵処理部 8 を持つ証明装置の例を図 6 に基づいて説明している。図 6 は秘密鍵処理部 8 の詳細な構成図である。

【0 1 4 9】

秘密鍵処理部 8 は、実施例 4 の証明装置における秘密鍵処理部 8 の構成と同様である。

【0 1 5 0】

レスポンス生成部 1 0 は、乱数生成部 1 2 の生成した乱数 k とコミットメント生成部 1 3 の生成したコミットメント w とチャレンジ入力部 9 に入力されたチャレンジ c とハッシュ値算出部 6 が生成したハッシュ値 $X(M)$ よりレスポンス r を生成する。

【0 1 5 1】

以下、レスポンス r の具体的な生成方法を、DSA 認証を例にとって説明しよう。

【0 1 5 2】

ここでは、有限群 G は離散対数問題が困難であるとする。また、ハッシュ値算出部 6 が生成するハッシュ値 $X(M)$ と乱数生成部 1 2 が生成する乱数 k とチャレンジ入力部 9 に入力されるチャレンジ c は p 元体 F_p の元である。

【0 1 5 3】

[DSA 認証の場合]

コミットメント w は

【0 1 5 4】

【数 3 2】

$$w = \pi(k, g)$$

として生成され、レスポンス r は

【0 1 5 5】

【数 3 3】

$$r = (c - wX(M)) / k$$

として生成される。

【0 1 5 6】

ちなみに、レスポンス r を

【0 1 5 7】

【数 3 4】

$$r \leftarrow (w, r)$$

としてコミットメント w とレスポンス r の対 (w, r) で置き換えると、レスポンス r はチャレンジ c に対する DSA 署名に他ならない。

【0 1 5 8】

[実施例 6 : 証明器発行装置]

本実施例では、固有値 d を入力すると、固有値 d に依存したハッシュ関数 X を備えた証明器（実施例 2 ないし 5）を発行する装置を図 7 に基づいて説明する。

図 7 は証明器発行装置の構成図である。本実施例の証明器発行装置は、ハッシュ値生成装置（実施例 1）において用いたものと等価な固有値入力部 1 と関数生成固有値記憶部 3 と値生成固有値算出部 4 を持つ。

【0159】

固有値入力部 1 は、ハッシュ関数 X の生成に要するパラメータである固有値 d が入力される。関数生成固有値記憶部 3 は、値生成固有値の生成に要するパラメータである関数生成固有値 s を保持する。値生成固有値算出部 4 は、関数生成固有値記憶部 3 に保持された関数生成固有値 s と固有値入力部 1 に入力された固有値 d より値生成固有値 u を生成する。値生成固有値書込部 15 は、値生成固有値算出部 4 が生成した値生成固有値 u を証明器 T に書き込む。証明器発行部 16 は、値生成固有値書込部 15 が値生成固有値 u を書き込んだ証明器 T を発行する。

【0160】

〔実施例 7：証明証型認証装置（2 方向認証）〕

本実施例では、証明器（実施例 2, 3）を用いた認証装置を図 8 と図 9 に基づいて説明する。図 8 は証明証型認証装置の構成図であり、図 9 は秘密鍵処理検証部の詳細な構成図である。

【0161】

証明証記憶部 17 は、証明器に固有のハッシュ関数 X によるメッセージ M のハッシュ値 $X(M)$ を秘密鍵と見なしたものと対をなす公開鍵 y を証明する証明証 C を保持する。証明証検証部 18 は、証明証記憶部 17 に保持された証明証 C を検証し、検証に成功すれば、証明証 C が証明する公開鍵 y を取得する。秘密鍵処理検証部 19 は、証明証検証部 18 が取得した公開鍵 y に基づいて、証明器と対話をすることによって証明器の秘密鍵処理を検証する。

【0162】

この認証装置は、例えば、スマートカードリーダー内の ROM に焼き付けられたプログラムとして実現しても良い。

【0163】

また、この認証装置は、例えば、アクセス制御が施されるアプリケーションプログラム内に検出が困難な様に埋入されたコードとして実現してもよい。アプリ

ケーションプログラムは実行時の要所要所でこのコードによって認証を行うことによって、ユーザーの利用資格を検証する。

【0164】

また、この認証装置は、例えば、アクセス制御が施されるデジタルコンテンツをレンダリングするアプリケーションプログラム内に検出が困難な様に埋入されたコードとして実現してもよい。保護されたコンテンツをレンダリングする場合、アプリケーションプログラムは実行時の要所要所でこのコードによって認証を行うことによって、ユーザーの利用資格を検証する。

【0165】

証明証検証部 18 は、保持された証明証 C に基づいた処理を行ってもよい。例えば、証明証 C には有効期限情報等の利用条件が記載されており、利用条件が満たされない場合は証明証の検証に失敗するように構成してもよい。このように構成することで、証明証に有効期限情報等の利用条件を対応づけることが可能となる。

【0166】

また、例えば、証明証 C には認証の識別子が記載されており、その識別子が予見されるものと異なっている場合は証明証の検証に失敗するように構成してもよい。このように構成することで、証明証の正しさを保証する証明証発行者の署名鍵を固定したまま、複数の認証に対応することができる。

【0167】

秘密鍵処理検証部 19 は、チャレンジ生成部 20 とレスポンス検証部 21 より構成される。

【0168】

チャレンジ生成部 20 は、認証のための質疑情報であるチャレンジ c を生成する。レスポンス検証部 21 は、証明証検証部 18 より取得した公開鍵 y とチャレンジ生成部 20 が生成したチャレンジ c を用いて証明器のレスポンス出力部 11 より取得したレスポンス r を検証する。

【0169】

チャレンジ生成部 20 はチャレンジ c をランダムに生成しても良いし、ハッシュ

ハッシュ関数 h を用いて署名を生成したいメッセージ m のハッシュ値として $c = h(m)$ として生成しても良いし、復号したい暗号文 K を選び $c = K$ としても良いし、乱数 k を生成し、復号したい暗号文 K に乱数 k でブラインド効果を付与することによって生成しても良い。

【0170】

以下、レスポンス r の検証方法を、様々なバリエーションを交えて説明しよう。

【0171】

証明証検証部 18 より取得される公開鍵 y は、DSA 署名、ElGamal 署名の変種、Nyberg-Rueppel 署名、Schnorr 署名、メッセージ復元型 Guillou-Quisquater 署名、Guillou-Quisquater 署名の例では有限群 G の元であり、RSA の例では有限環 Λ の元である (RSA の場合、公開鍵 y は実質的には整数と見なしている)。

【0172】

[DSA 署名の場合]

レスポンス $r = (r_0, r_1)$ の検証式は

【0173】

【数 35】

$$r_0 = \pi((c / r_1) \cdot g + (r_0 / r_1) \cdot y)$$

である。

【0174】

[変形 ElGamal 署名 1 の場合]

レスポンス $r = (r_0, r_1)$ の検証式は

【0175】

【数 36】

$$r_0 = \pi((r_1 / c) \cdot g + (r_0 / c) \cdot y)$$

である。

【0176】

[変形 ElGamal 署名 2 の場合]

レスポンス $r = (r_0, r_1)$ の検証式は

【0177】

【数37】

$$r_0 = \pi((r_1 / r_0) \cdot g + (c / r_0) \cdot y)$$

である。

【0178】

[変形 El Gamal 署名3の場合]

レスポンス $r = (r_0, r_1)$ の検証式は

【0179】

【数38】

$$r_0 = \pi((r_0 / r_1) \cdot g + (c / r_1) \cdot y)$$

である。

【0180】

[変形 El Gamal 署名4の場合]

レスポンス $r = (r_0, r_1)$ の検証式は

【0181】

【数39】

$$r_0 = \pi((r_0 / c) \cdot g + (r_1 / c) \cdot y)$$

である。

【0182】

[変形 El Gamal 署名5の場合]

レスポンス $r = (r_0, r_1)$ の検証式は

【0183】

【数40】

$$r_0 = \pi((c / r_0) \cdot g + (r_1 / r_0) \cdot y)$$

である。

【0184】

[Nyberg-Rueppel 署名の場合]

レスポンス $r = (r_0, r_1)$ の検証式は

【0185】

【数41】

$$c = r_0 + \pi(r_1 \cdot g + r_0 \cdot y)$$

である。

【0186】

[Schnorr署名]

レスポンス $r = (r_0, r_1)$ の検証式は

【0187】

【数42】

$$r_0 = h(c \parallel \varepsilon(r_1 \cdot g + r_0 \cdot y))$$

である。

【0188】

[RSAの場合]

レスポンス r の検証式は

【0189】

【数43】

$$c = y \cdot r$$

である。

【0190】

[メッセージ復元型 Guillou-Quisquater 署名の場合]

レスポンス $r = (r_0, r_1)$ の検証式は

【0191】

【数44】

$$c = r_0 + \pi(p \cdot r_1 + y \cdot r_0)$$

である。

【0192】

[Guillou-Quisquater 署名の場合]

レスポンス $r = (r_0, r_1)$ の検証式は

【0193】

【数 45】

$$r_0 = h(c \mid \varepsilon(p \cdot r_1 + y \cdot r_0))$$

である。

【0194】

【実施例 8：証明証型認証装置（2 方向認証 Diffie-Hellman 鍵共有）】

本実施例では、実施例 7 と異なる秘密鍵処理検証部 19 を持つ認証装置の例を図 10 に基づいて説明している。図 10 は秘密鍵処理検証部 19 の詳細な構成図である。

【0195】

秘密鍵処理検証部 19 は、乱数生成部 22 とチャレンジ生成部 20 とレスポンス検証部 21 より構成される。

【0196】

乱数生成部 22 は乱数 k を生成する。チャレンジ生成部 20 は、乱数生成部 22 の生成した乱数 k より認証のための質疑情報であるチャレンジ c を生成する。レスポンス検証部 21 は、証明証検証部 18 より取得した公開鍵 y と乱数生成部 22 が生成した乱数 k より証明器のレスポンス出力部 11 から取得したレスポンス r を検証する。

【0197】

以下、チャレンジ c の生成方法とレスポンス r の検証式を、Diffie-Hellman 鍵共有を例にとって具体的に説明しよう。

【0198】

証明証検証部 18 より取得される公開鍵 y および乱数生成部 22 で生成される乱数 k は有限群 G の元である。

【0199】

【Diffie-Hellman 鍵共有の場合】

チャレンジ c は

【0200】

【数 46】

$$c = k \cdot g$$

として生成される。レスポンス r の検証式は

$$[0201]$$

【数 4 7】

$$k \cdot y = r$$

である。

$$[0202]$$

【実施例 9：証明証型認証装置（3 方向認証）】

本実施例では、証明器（実施例 4）を用いた認証装置を図 8 と図 1 1 に基づいて説明する。図 8 は証明証型認証装置の構成図であり、図 1 1 は秘密鍵処理検証部の詳細な構成図である。

$$[0203]$$

秘密鍵処理検証部 1 9 がチャレンジ生成部 2 0 とレスポンス検証部 2 1 より構成されるのは実施例 7 と同様である。

$$[0204]$$

チャレンジ生成部 2 0 は、認証のための質疑情報であるチャレンジ c をランダムに生成する。レスポンス検証部 2 1 は、証明証検証部 1 8 より取得した公開鍵 y とチャレンジ c の生成に先立って証明器のコミットメント出力部 1 1 より取得したコミットメント w とチャレンジ生成部 2 2 が生成したチャレンジ c を用いて証明器のレスポンス出力部 1 1 より取得したレスポンス r を検証する。

$$[0205]$$

以下、レスポンス r の検証方法を Schnorr 認証と Guillou-Quisquater 認証を例にとって具体的に説明しよう。

$$[0206]$$

証明証検証部 1 8 より取得される公開鍵 y は有限群 G の元である。

$$[0207]$$

【Schnorr 認証の場合】

レスポンス r の検証式は

$$[0208]$$

【数 48】

$$w = \pi (r \cdot g + c \cdot y)$$

である。

【0209】

[Guillou-Quisquater 認証の場合]

レスポンス r の検証式は

【0210】

【数 49】

$$w = \pi (p \cdot r + c \cdot y)$$

である。

【0211】

[Fiat-Shamir 認証の場合]

レスポンス r の検証式は

【0212】

【数 50】

$$w = \pi (p \cdot r + \sum c_i \cdot y_i)$$

である。

【0213】

[実施例 10：証明証発行装置]

本実施例では、固有价值 d とメッセージ M を入力すると、ハッシュ値生成器（実施例 1）を用いて、固有价值 d に対応づけて発行された証明器（実施例 2 ないし 6）を用いた証明証型認証装置（実施例 7 ないし 9）で利用される証明証 C を発行する装置を、図 10 に基づいて説明する。図 10 は証明証発行装置の構成図である。

【0214】

公開鍵算出部 23 は、ハッシュ値生成器の生成したハッシュ値 $X(M)$ を秘密鍵と見なし、対をなす公開鍵 y を算出する。証明証発行部 24 は、公開鍵算出部 23 が算出した公開鍵 y を証明する証明証 C を発行する。

【0215】

証明証発行部 2 4 は、メッセージ M に認証の識別子 a i d を記載するように構成しても良い。このようにすることで異なる認証の識別子に対応するハッシュ値 $X(M)$ が確実に異なるものにすることができる。

【0 2 1 6】

以下、公開鍵 y の算出方法を具体的に説明する。

【0 2 1 7】

[離散対数問題が困難な有限群 G の場合]

公開鍵 y は

【0 2 1 8】

【数 5 1】

$$y = X(M) \cdot g$$

として算出される。

【0 2 1 9】

[零化域決定問題が困難な有限群 G の場合 1]

Guillou-Quisquater 認証, Guillou-Quisquater 署名, メッセージ復元型 Guillou-Quisquater 署名では、公開鍵 y は

【0 2 2 0】

【数 5 2】

$$y = p \cdot X(M)$$

として算出される。

【0 2 2 1】

Fiat-Shamir 認証では、公開鍵 y は

【0 2 2 2】

【数 5 3】

$$y_i = 2 \cdot X(M)_i$$

として算出される。ただし、

【0 2 2 3】

【数 5 4】

$$y = (y_1, \dots, y_n)$$

$$X(M) = (X(M)_1, \dots, X(M)_n)$$

である。

【0 2 2 4】

〔零化域決定問題が困難な有限群 G の場合 2〕

R S A では、公開鍵 y は

【0 2 2 5】

【数 5 5】

$$y = X(M)^{-1}$$

として算出される。

【0 2 2 6】

〔実施例 1 1 : チケット型認証装置 (チャレンジを変換) 〕

本実施例では、証明器 (実施例 2, 3) を用いた認証装置を図 1 3 と図 1 4 に基づいて説明する。図 1 3 はアクセスチケット型認証装置の構成図である。本実施例の認証装置は、証明証型認証装置 (実施例 7 ないし 9) と等価な秘密鍵処理検証部 1 9 を持つ。図 1 4 は秘密鍵処理変換部 2 7 の詳細な構成図である。

【0 2 2 7】

公開鍵記憶部 2 5 は、公開鍵 y を保持する。アクセスチケット記憶部 2 6 は、公開鍵と対をなす秘密鍵とハッシュ値 $X(M)$ とから定まるアクセスチケット t を保持する。秘密鍵処理検証部 1 9 は、公開鍵記憶部 2 5 が保持する公開鍵 y に基づいて証明器と対話をすることによって証明器の秘密鍵処理を検証するが、この際、秘密鍵処理変換部 2 7 が、アクセスチケット記憶部 2 6 が保持するアクセスチケット t を用いて証明器の秘密鍵処理を変換する。

【0 2 2 8】

この認証装置は、例えば、スマートカードリーダー内の R O M に焼き付けられたプログラムとして実現しても良い。

【0 2 2 9】

また、この認証装置は、例えば、アクセス制御が施されるアプリケーションプログラム内に検出が困難な様に埋入されたコードとして実現してもよい。アプリ

ケーションプログラムは実行時の要所要所でこのコードによって認証を行うことによって、ユーザーの利用資格を検証する。

【0230】

秘密鍵処理検証部19は、少なくともチャレンジ生成部20を備え、秘密鍵処理部8は、少なくともチャレンジ入力部9を備える。

【0231】

秘密鍵処理変換部27は、チャレンジ更新部28より構成される。

【0232】

チャレンジ更新部28は、チャレンジ生成部20の生成したチャレンジcをアクセスチケット記憶部26が保持するアクセスチケットtを用いて更新し、更新されたチャレンジcをチャレンジ入力部9に入力する。

【0233】

以下、アクセスチケットtの定義方法とチャレンジcの更新方法を、Diffie-Hellman鍵共有と変形ElGamal署名3とSchnorr認証とRSAを例にとって説明しよう。

【0234】

秘密鍵xは、Diffie-Hellman鍵共有、変形ElGamal署名3、Schnorr認証の例ではp元体 F_p の元であり、RSAの例では有限環 Λ の元である（RSAの場合、対応するアクセスチケットtは実質的には整数と見なしている）。

【0235】

[Diffie-Hellman鍵共有の場合]

アクセスチケットtの定義式は

【0236】

【数56】

$$t = x / X(M)$$

であり、チャレンジcは

【0237】

【数57】

$$c \leftarrow t, c$$

として変換される。

【0 2 3 8】

[変形 E 1 G a m a 1 署名 3 の場合]

アクセスチケット t の定義式は

【0 2 3 9】

【数 5 8】

$$t = x / X (M)$$

であり, チャレンジ c は

【0 2 4 0】

【数 5 9】

$$c \leftarrow c t$$

として変換される。

【0 2 4 1】

[S c h n o r r 認証の場合]

アクセスチケット t の定義式は

【0 2 4 2】

【数 6 0】

$$t = x / X (M)$$

であり, チャレンジ c は

【0 2 4 3】

【数 6 1】

$$c \leftarrow c t$$

として変換される。

【0 2 4 4】

[R S A の場合]

アクセスチケット t の定義式は

【0 2 4 5】

【数 6 2】

$$t = x / X (M)$$

であり、チャレンジ c は

【 0 2 4 6 】

【 数 6 3 】

$$c \leftarrow t . c$$

として変換される。

【 0 2 4 7 】

〔 実施例 1 2 : チケット型認証装置 (レスポンスを変換) 〕

本実施例では、実施例 1 1 と異なる秘密鍵処理変換部 2 7 を持つ認証装置の例を図 1 5 に基づいて説明している。図 1 5 は秘密鍵処理変換部 2 7 の詳細な構成図である。

【 0 2 4 8 】

秘密鍵処理部 8 は、少なくともレスポンス出力部 1 1 を備える。

【 0 2 4 9 】

秘密鍵処理変換部 2 7 は、レスポンス更新部 2 9 より構成される。

【 0 2 5 0 】

レスポンス更新部 2 9 は、レスポンス出力部 1 1 の出力したレスポンス r をアクセスチケット記憶部 2 6 が保持するアクセスチケット t を用いて更新する。

【 0 2 5 1 】

以下、アクセスチケット t の定義方法とレスポンス r の具体的な更新方法を、Diffie-Hellman 鍵共有と変形 ElGamal 署名 1 と変形 ElGamal 署名 4 と変形 ElGamal 署名 5 と RSA と Nyberg-Rueppel 署名とメッセージ復元型 Guillou-Quisquater 署名を例にとって説明しよう。

【 0 2 5 2 】

秘密鍵 x は、Diffie-Hellman 鍵共有、変形 ElGamal 署名 1、変形 ElGamal 署名 4、変形 ElGamal 署名 5 の例では p 元体 F_p の元であり、RSA の例では有限環 Λ の元である (RSA の場合、対応するアクセスチケット t は実質的には整数と見なしている)。

【0 2 5 3】

[D i f f i e - H e l l m a n 鍵共有の場合]

アクセスチケット t の定義式は

【0 2 5 4】

【数 6 4】

$$t = x / X (M)$$

であり、レスポンス r は

【0 2 5 5】

【数 6 5】

$$r \leftarrow t \cdot r$$

として変換される。

【0 2 5 6】

[変形 E l G a m a l 署名 1 の場合]

アクセスチケット t の定義式は

【0 2 5 7】

【数 6 6】

$$t = x - X (M)$$

であり、レスポンス $r = (r_0, r_1)$ は

【0 2 5 8】

【数 6 7】

$$r_1 \leftarrow r_1 - r_0 t$$

として変換される。

【0 2 5 9】

[変形 E l G a m a l 署名 4 の場合]

アクセスチケット t の定義式は

【0 2 6 0】

【数 6 8】

$$t = x / X (M)$$

であり、レスポンス $r = (r_0, r_1)$ は

【0 2 6 1】

【数 6 9】

$$r_1 \leftarrow r_1 / t$$

として変換される。

【0 2 6 2】

[変形 E 1 G a m a l 署名 5 の場合]

アクセスチケット t の定義式は

【0 2 6 3】

【数 7 0】

$$t = x / X (M)$$

であり、レスポンス $r = (r_0, r_1)$ は

【0 2 6 4】

【数 7 1】

$$r_1 \leftarrow r_1 / t$$

として変換される。

【0 2 6 5】

[R S A の場合]

アクセスチケット t の定義式は

【0 2 6 6】

【数 7 2】

$$t = x / X (M)$$

であり、チャレンジ c は

【0 2 6 7】

【数 7 3】

$$r \leftarrow t \cdot r$$

として変換される。

【0 2 6 8】

[N y b e r g - R u e p p e l 署名の場合]

アクセスチケット t の定義式は

【0 2 6 9】

【数 7 4】

$$t = x - X(M)$$

であり、レスポンス $r = (r_0, r_1)$ は

【0 2 7 0】

【数 7 5】

$$r_1 \leftarrow r_1 - r_0 t$$

として変換される。

【0 2 7 1】

[メッセージ復元型Guillou-Quisquater署名の場合]

アクセスチケット t の定義式は

【0 2 7 2】

【数 7 6】

$$t = x - X(M)$$

であり、レスポンス $r = (r_0, r_1)$ は

【0 2 7 3】

【数 7 7】

$$r_1 \leftarrow r_1 - r_0 \cdot t$$

として変換される。

【0 2 7 4】

[実施例 1 3 : チケット型認証装置 (チャレンジでレスポンスを変換)]

本実施例では、実施例 1 1 ないし 1 2 と異なる秘密鍵処理変換部 2 7 を持つ認証装置の例を図 1 6 に基づいて説明している。図 1 6 は秘密鍵処理変換部 2 7 の詳細な構成図である。

【0 2 7 5】

秘密鍵処理検証部 1 9 は、少なくともチャレンジ生成部 2 0 を備え、秘密鍵処理部 8 は、少なくともレスポンス出力部 1 1 を備える。

【0 2 7 6】

秘密鍵処理変換部 2 7 は、レスポンス更新部 2 9 より構成される。

【0277】

レスポンス更新部 29 は、レスポンス出力部 11 の出力したレスポンス r をアクセスチケット記憶部 26 が保持するアクセスチケット t とチャレンジ生成部 20 が生成したチャレンジ c を用いて更新する。

【0278】

以下、アクセスチケット t の定義方法とレスポンス r の更新方法を、Diffie-Hellman 鍵共有、RSA、変形 ElGamal 署名 2、Schnorr 認証、Guillou-Quisquater 認証を例にとって説明しよう。

【0279】

秘密鍵 x は、Diffie-Hellman 鍵共有、変形 ElGamal 署名 1、変形 ElGamal 署名 4、変形 ElGamal 署名 5 の例では p 元体 F_p の元であり、RSA の例では有限環 Λ の元である（RSA の場合、対応するアクセスチケット t は実質的には整数と見なしている）。

【0280】

[Diffie-Hellman 鍵共有の場合]

アクセスチケット t の定義式は

【0281】

【数 78】

$$t = x - X(M)$$

であり、レスポンス r は

【0282】

【数 79】

$$r \leftarrow r + t \cdot c$$

として変換される。

【0283】

[RSA の場合]

アクセスチケット t の定義式は

【0284】

【数 8 0】

$$t = x - X (M)$$

であり、レスポンス r は

【0 2 8 5】

【数 8 1】

$$r \leftarrow r + t \cdot c$$

として変換される。

【0 2 8 6】

[Schnorr 認証の場合]

アクセスチケット t の定義式は

【0 2 8 7】

【数 8 2】

$$t = x - X (M)$$

であり、レスポンス r は

【0 2 8 8】

【数 8 3】

$$r \leftarrow r + c \cdot t$$

として変換される。

【0 2 8 9】

[Guillou-Quisquater 認証の場合]

アクセスチケット t の定義式は

【0 2 9 0】

【数 8 4】

$$t = x - X (M)$$

であり、レスポンス r は

【0 2 9 1】

【数 8 5】

$$r \leftarrow r + c \cdot t$$

として変換される。

【0292】

[実施例 14 : チケット型認証装置 (更新済みチャレンジでレスポンスを変換)
]

本実施例では、実施例 11 ないし 13 と異なる秘密鍵処理変換部 27 を持つ認証装置の例を図 17 に基づいて説明している。図 17 は秘密鍵処理変換部 27 の詳細な構成図である。

【0293】

秘密鍵処理検証部 19 は、少なくともチャレンジ生成部 20 を備え、秘密鍵処理部 8 は、少なくともコミットメント出力部 11 とチャレンジ入力部 9 とレスポンス出力部 11 を備える。

【0294】

秘密鍵処理変換部 27 は、チャレンジ更新部 28 とレスポンス更新部 29 より構成される。

【0295】

チャレンジ更新部 28 は、チャレンジ生成部 20 が生成したチャレンジ c をコミットメント出力部 14 より取得したコミットメント w を用いて更新し、チャレンジ入力部 9 に更新されたチャレンジ c を入力する。

【0296】

レスポンス更新部 29 は、レスポンス出力部 11 の出力したレスポンス r をアクセスチケット記憶部 26 が保持するアクセスチケット t とチャレンジ更新部 28 が更新したチャレンジ c を用いて更新する。

【0297】

以下、アクセスチケット t の定義方法とチャレンジ c およびレスポンス r の更新方法を、Schnorr 認証と Guillou-Quisquater 認証を例にとって説明しよう。

【0298】

秘密鍵 x は、Schnorr 認証の例では p 元体 F_p の元であり、Guillou-Quisquater 認証の例では有限群 G の元である。

【0299】

[Schnorr 認証 1 から Nyberg-Rueppel 署名への変換の場合]

アクセスチケット t の定義式は

$$[0300]$$

【数 86】

$$t = x - X(M)$$

であり, チャレンジ c は

$$[0301]$$

【数 87】

$$c \leftarrow c - w$$

として変換され, レスポンス r は

$$[0302]$$

【数 88】

$$r \leftarrow r + ct$$

として変換される。

$$[0303]$$

[Schnorr 認証 2 から Schnorr 署名への変換の場合]

アクセスチケット t の定義式は

$$[0304]$$

【数 89】

$$t = x - X(M)$$

であり, チャレンジ c は

$$[0305]$$

【数 90】

$$c \leftarrow \eta(c \parallel \varepsilon(w))$$

として変換され, レスポンス r は

$$[0306]$$

【数 91】

$$r \leftarrow r + ct$$

として変換される。

【0307】

[Guillou-Quisquater 認証1からメッセージ復元型 Guillou-Quisquater 署名への変換の場合]

アクセスチケット t の定義式は

【0308】

【数92】

$$t = x - X(M)$$

であり、チャレンジ c は

【0309】

【数93】

$$c \leftarrow c - w$$

として変換され、レスポンス r は

【0310】

【数94】

$$r \leftarrow r + ct$$

として変換される。

【0311】

[Guillou-Quisquater 認証2から Guillou-Quisquater 署名への変換の場合]

アクセスチケット t の定義式は

【0312】

【数95】

$$t = x - X(M)$$

であり、チャレンジ c は

【0313】

【数96】

$$c \leftarrow \eta(c \parallel \varepsilon(w))$$

として変換され、レスポンス r は

【0 3 1 4】

【数 9 7】

$$r \leftarrow r + c t$$

として変換される。

【0 3 1 5】

〔実施例 1 5：チケット型認証装置（コミットメントでチャレンジを変換）〕

本実施例では、実施例 1 1 ないし 1 4 と異なる秘密鍵処理変換部 2 7 を持つ認証装置の例を図 1 8 に基づいて説明している。図 1 8 は秘密鍵処理変換部 2 7 の詳細な構成図である。

【0 3 1 6】

秘密鍵処理検証部 1 9 は、少なくともチャレンジ生成部 2 0 を備え、秘密鍵処理部 8 は、少なくともコミットメント出力部 1 1 とチャレンジ入力部 9 とレスポンス出力部 1 1 を備える。

【0 3 1 7】

秘密鍵処理変換部 2 7 は、チャレンジ更新部 2 8 とレスポンス更新部 2 9 より構成される。

【0 3 1 8】

チャレンジ更新部 2 8 は、チャレンジ生成部 2 0 が生成したチャレンジ c をアクセスチケット記憶部 2 6 が保持するアクセスチケット t とコミットメント出力部 1 4 より取得したコミットメント w を用いて更新し、チャレンジ入力部 9 に更新されたチャレンジ c を入力する。

【0 3 1 9】

レスポンス更新部 2 9 は、レスポンス出力部 1 1 の出力したレスポンス r をコミットメント出力部 1 4 より取得したコミットメント w を用いて更新する。

【0 3 2 0】

以下、アクセスチケット t の定義方法とチャレンジ c およびレスポンス r の更新方法を、DSA 認証を例にとって説明しよう。

【0 3 2 1】

秘密鍵 x は p 元体 F_p の元である。

【0 3 2 2】

[D S A 認証から D S A 署名への変換の場合]

アクセスチケット t の定義式は

【0 3 2 3】

【数 9 8】

$$t = x - X(M)$$

であり、チャレンジ c は

【0 3 2 4】

【数 9 9】

$$c \leftarrow c + w t$$

として変換され、レスポンス r は

【0 3 2 5】

【数 1 0 0】

$$r \leftarrow (w, r)$$

として変換される。

【0 3 2 6】

[実施例 1 6 : チケット発行装置]

本実施例では、固有値 d とメッセージ M を入力すると、ハッシュ値生成器（実施例 1）を用いて、固有値 d に対応づけて発行された証明器（実施例 2 ないし 6）を用いたアクセスチケット型認証装置（実施例 1 1 ないし 1 5）で利用されるアクセスチケット t を発行する装置を、図 1 9 に基づいて説明する。図 1 9 はアクセスチケット発行装置の構成図である。

【0 3 2 7】

アクセスチケット算出部 3 1 は、ハッシュ値生成器の生成したハッシュ値 $X(M)$ と秘密鍵記憶部 3 0 に保持された秘密鍵 x よりアクセスチケット t を算出する。アクセスチケット発行部 3 2 は、アクセスチケット算出部 3 1 が算出したアクセスチケット t を発行する。

【0 3 2 8】

メッセージ M に認証の識別子 $a i d$ を含ませるように構成しても良い。このよ

うにすることで異なる認証の識別子に対応するハッシュ値 $X(M)$ が確実に異なるものにすることができる。

【0 3 2 9】

以下、アクセスチケット t の算出方法を具体的に説明する。

【0 3 3 0】

[秘密鍵が p 元体 F_p の元の場合]

アクセスチケット t は

【0 3 3 1】

【数 1 0 1】

$$t = x - X(M)$$

あるいは

$$t = x / X(M)$$

として算出される。

【0 3 3 2】

[秘密鍵が有限群 G の元の場合]

アクセスチケット t は

【0 3 3 3】

【数 1 0 2】

$$t = x - X(M)$$

として算出される。

【0 3 3 4】

[秘密鍵が有限環 Λ の元の場合]

アクセスチケット t は

【0 3 3 5】

【数 1 0 3】

$$t = x - X(M)$$

あるいは

$$t = x / X(M)$$

として算出される。

【0 3 3 6】

[実施例 1 7 : 認証システム]

本実施例では、証明証型認証装置（実施例 7 ないし 9）あるいはアクセスチケット型認証装置（実施例 1 1 ないし 1 5）を用いた認証システムを図 2 0 に基づいて説明する。図 2 0 は認証システムの概念構成図である。

【0 3 3 7】

資格発行者（センター）3 3 は、ケーパビリティの発行器（実施例 1 0 あるいは 1 6）3 4 と、資格付与者に対応づけられた固有値 d を保持している。

【0 3 3 8】

資格付与者（ユーザー）3 5 は、証明器発行装置（実施例 6）によって固有値 d に対応づけられて発行された秘密ハッシュ関数 H を内部に持つ証明器（実施例 2 ないし 5）3 6 を保持している。

【0 3 3 9】

資格発行者 3 3 が資格付与者 3 5 に資格を付与する場合、メッセージ M に対応づけて資格の保持を表象するケーパビリティ x を、ケーパビリティの発行装置を用いて資格付与者 3 5 に発行する。

【0 3 4 0】

資格付与者 3 5 が資格検証者 3 7 に資格が付与されたことを証明する場合、ケーパビリティ x と証明装置を含む認証装置を用いる。

【0 3 4 1】

証明証型認証システムの場合、認証のケーパビリティは、トークン内の秘密ハッシュ値に対応する公開鍵と、認証の識別子を含み、資格検証者が信頼できる者の署名が施された証明証としてユーザーに与えられる。証明証の発行者は、証明証発行装置（実施例 1 0）を用いて固有値 d に対応するトークンの保持者たるユーザーにメッセージ M に対応づけて証明証を発行することができる。固有値としては、例えば、トークン識別子 $t i d$ を、メッセージ M としては、例えば、認証識別子 $a i d$ を選ぶことができる。

【0 3 4 2】

アクセスチケット型認証システムの場合、認証のケーパビリティは、トークン

内の秘密ハッシュ値と認証の識別子に対応する秘密鍵から計算される量であるアクセスチケットとしてユーザーに与えられる。アクセスチケットの発行者は、アクセスチケット発行装置（実施例 1 6）を用いて固有値 d に対応するトークンの保持者たるユーザーにメッセージ M に対応づけてアクセスチケットを発行することができる。固有値としては、例えば、トークン識別子 $t i d$ を、メッセージ M としては、例えば、認証識別子 $a i d$ を選ぶことができる。

【0 3 4 3】

【発明の効果】

以上、説明してきたように、本発明の一方向性関数生成技術およびそれに基づいた認証手法では、利用者が単一のトークンを備えることによって、単純な原理で多彩なケーパビリティを利用可能となる。

【図面の簡単な説明】

- 【図 1】 ハッシュ値生成装置の構成を示す図である。
- 【図 2】 証明装置の構成を示す図である。
- 【図 3】 秘密鍵処理部の詳細な構成を示す図である。
- 【図 4】 秘密鍵処理部の詳細な構成を示す図である。
- 【図 5】 秘密鍵処理部の詳細な構成を示す図である。
- 【図 6】 秘密鍵処理部の詳細な構成を示す図である。
- 【図 7】 証明器発行装置の構成を示す図である。
- 【図 8】 証明証型認証装置の構成を示す図である。
- 【図 9】 秘密鍵処理検証部の詳細な構成を示す図である。
- 【図 1 0】 秘密鍵処理検証部の詳細な構成を示す図である。
- 【図 1 1】 秘密鍵処理検証部の詳細な構成を示す図である。
- 【図 1 2】 証明証発行装置の構成を示す図である。
- 【図 1 3】 アクセスチケット型認証装置の構成を示す図である。
- 【図 1 4】 秘密鍵処理変換部の詳細な構成を示す図である。
- 【図 1 5】 秘密鍵処理変換部の詳細な構成を示す図である。
- 【図 1 6】 秘密鍵処理変換部の詳細な構成を示す図である。
- 【図 1 7】 秘密鍵処理変換部の詳細な構成を示す図である。

【図 1 8】 秘密鍵処理変換部の詳細な構成を示す図である。

【図 1 9】 アクセスチケット発行装置の構成を示す図である。

【図 2 0】 認証システムの概念構成を示す図である。

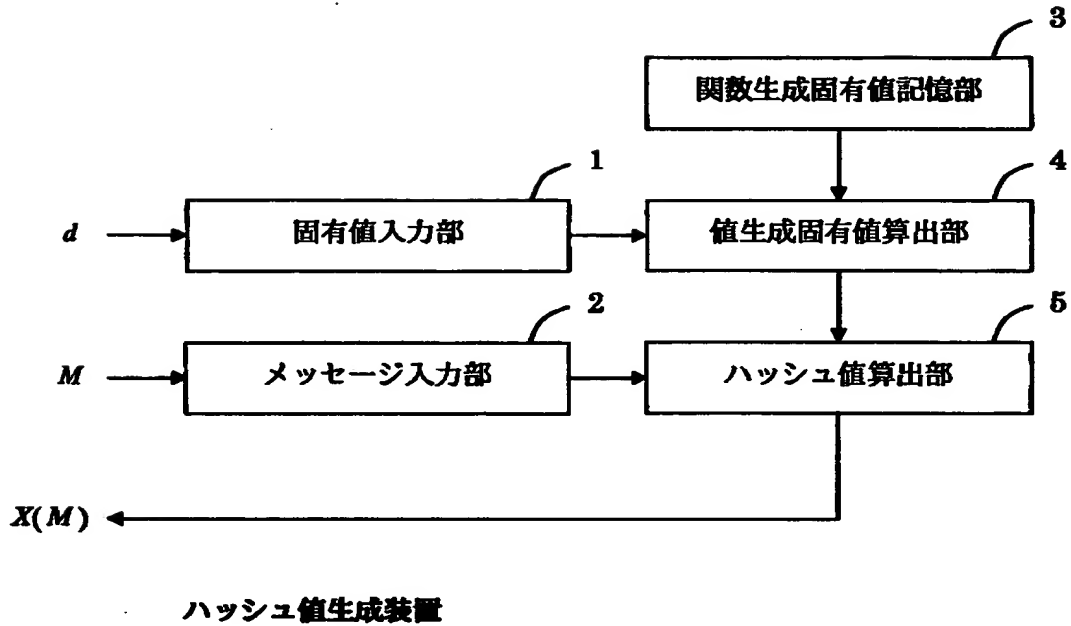
【符号の説明】

- 1 固有値入力部
- 2 メッセージ入力部
- 3 関数生成固有値記憶部
- 4 値生成固有値算出部
- 5 ハッシュ値算出部
- 6 ハッシュ値出力部
- 7 値生成固有値記憶部
- 8 秘密鍵処理部
- 9 チャレンジ入力部
- 1 0 レスポンス生成部
- 1 1 レスポンス出力部
- 1 2 乱数生成部
- 1 3 コミットメント生成部
- 1 4 コミットメント出力部
- 1 5 値生成固有値書込部
- 1 6 証明器発行部
- 1 7 証明証記憶部
- 1 8 証明証検証部
- 1 9 秘密鍵処理検証部
- 2 0 チャレンジ生成部
- 2 1 レスポンス検証部
- 2 2 乱数生成部
- 2 3 公開鍵算出部
- 2 4 証明証発行部
- 2 5 公開鍵記憶部

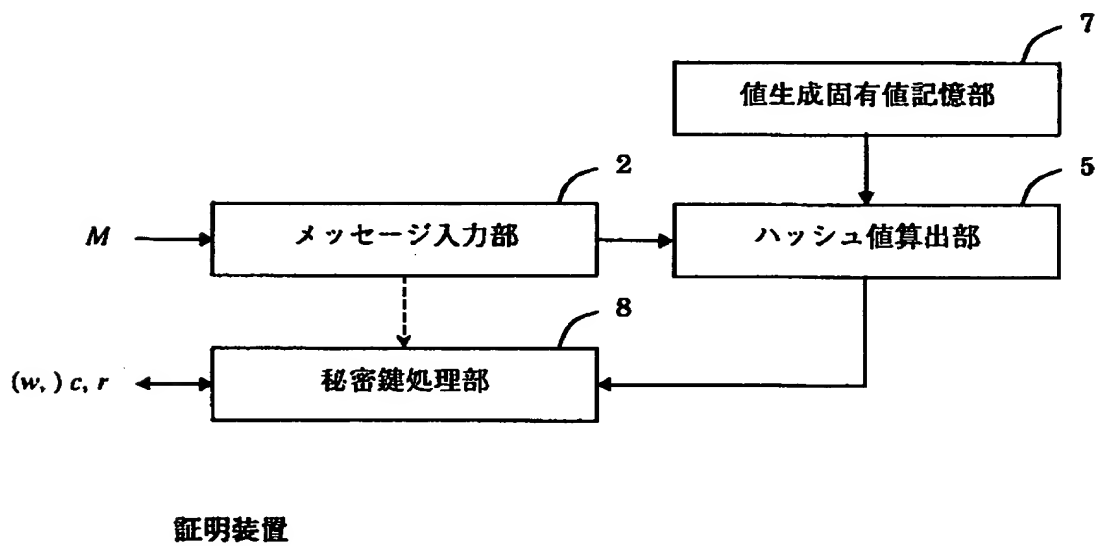
- 2 6 アクセスチケット記憶部
- 2 7 秘密鍵処理変換部
- 2 8 チャレンジ更新部
- 2 9 レスポンス更新部
- 3 0 秘密鍵記憶部
- 3 1 アクセスチケット算出部
- 3 2 アクセスチケット発行部
- 3 3 資格発行者
- 3 4 ケーパビリティ発行器
- 3 5 資格付与者
- 3 6 証明器
- 3 7 資格検証者

【書類名】 図面

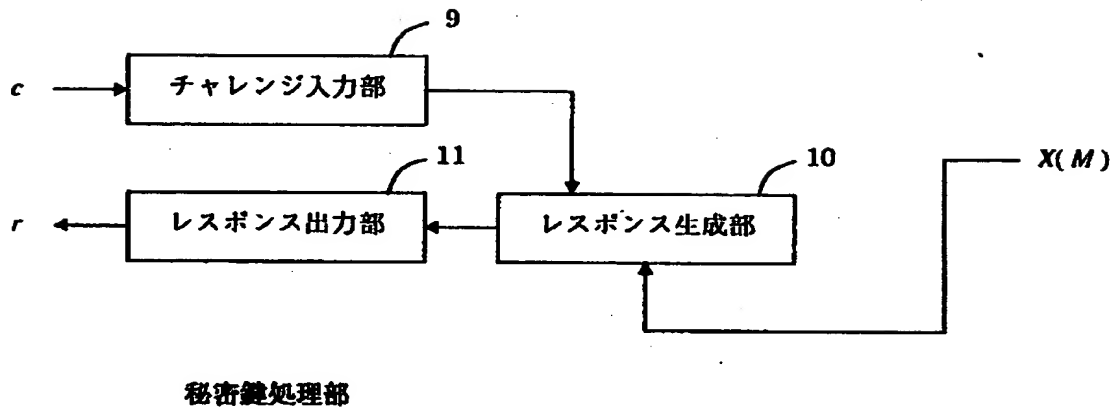
【図 1】



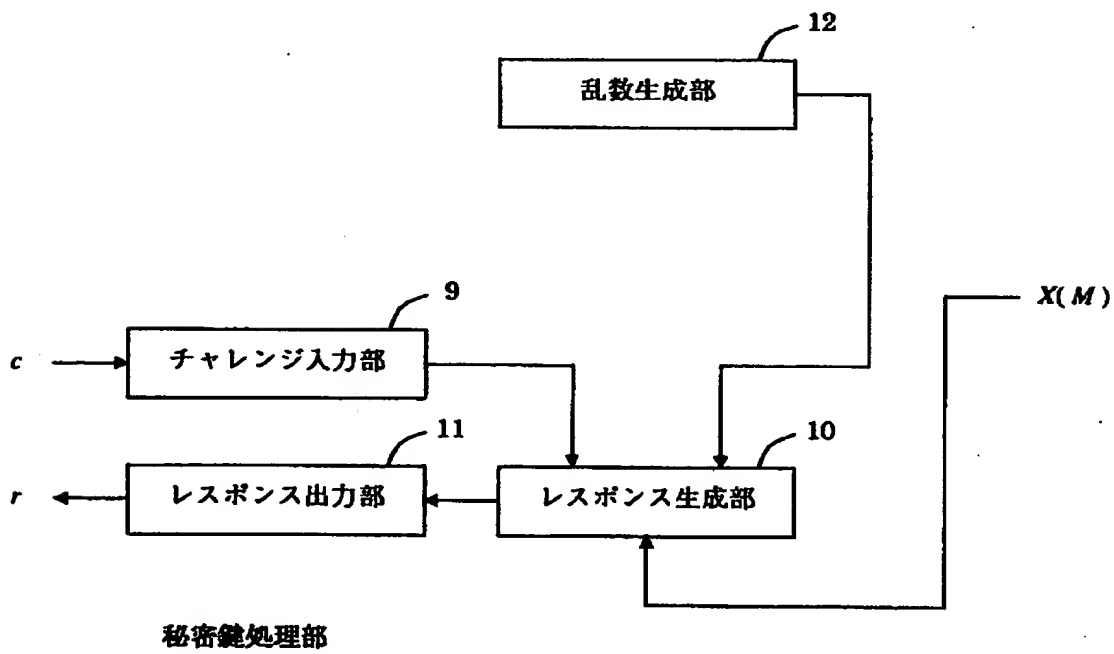
【図 2】



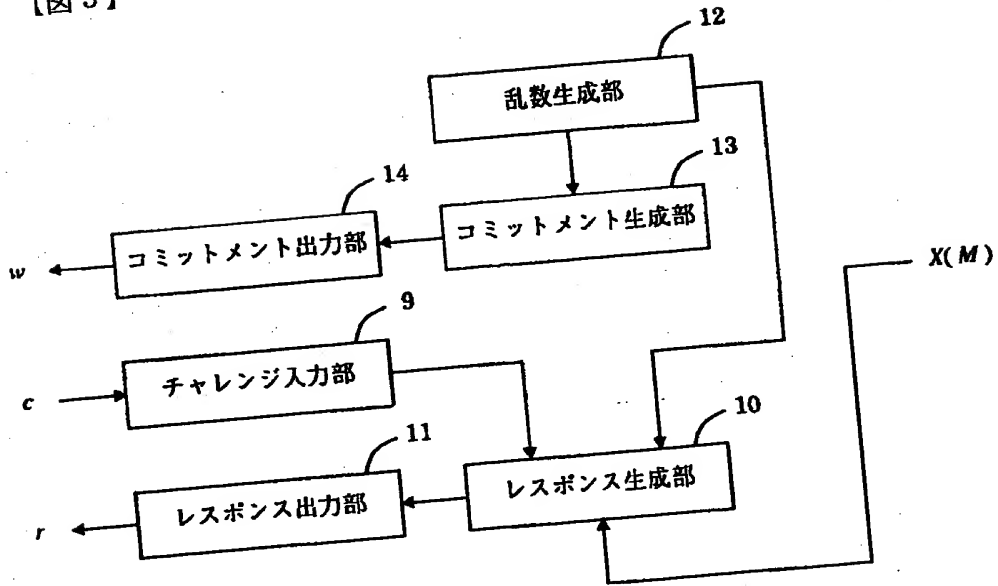
【図 3】



【図 4】

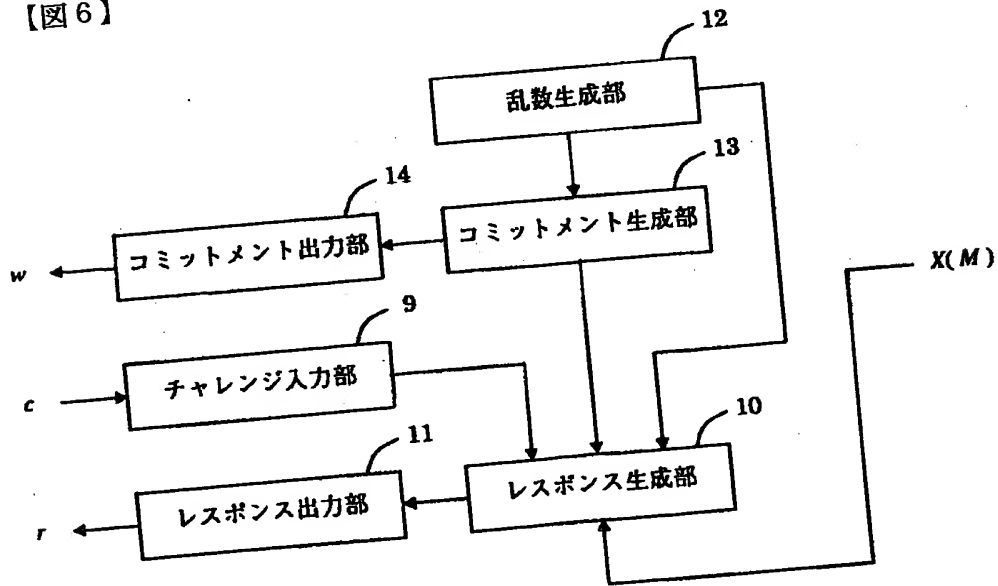


【図 5】



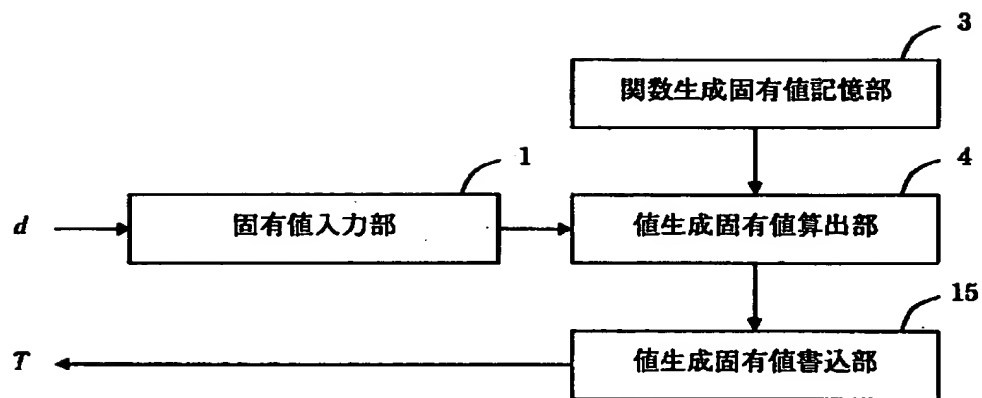
秘密鍵処理部

【図 6】



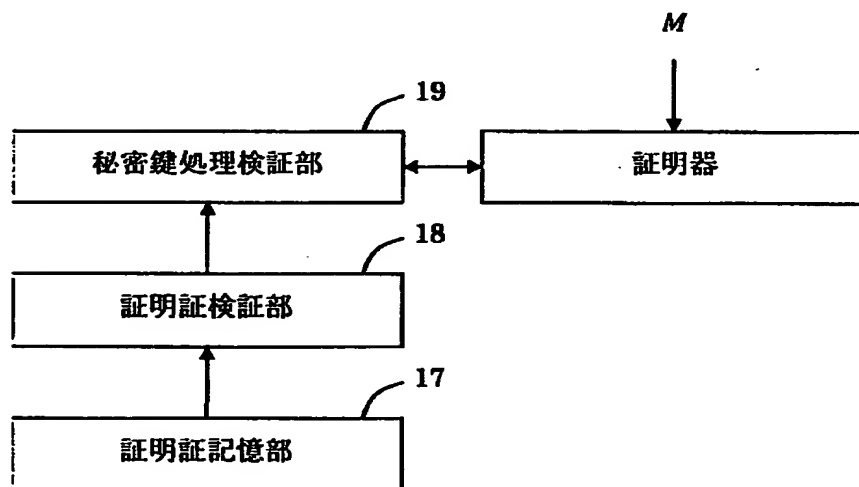
秘密鍵処理部

【図 7】



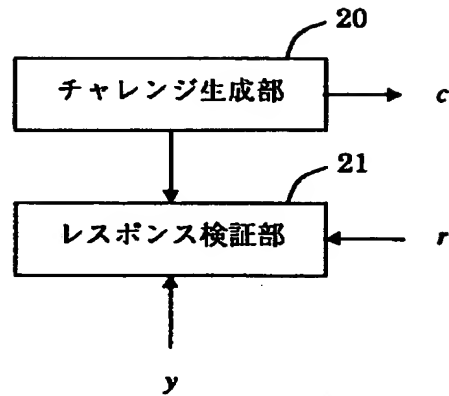
証明器発行装置

【図 8】



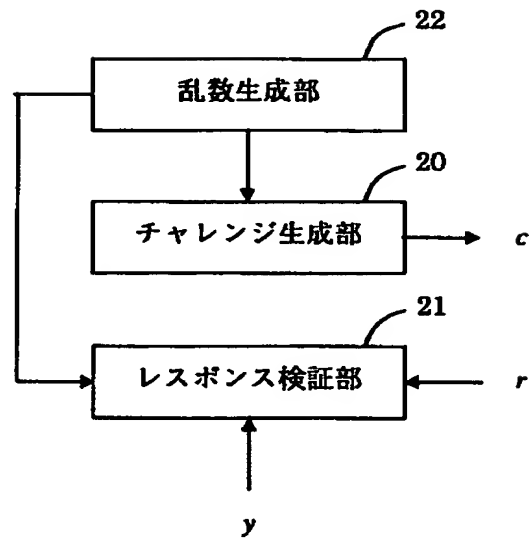
証明証型認証装置

【図 9】



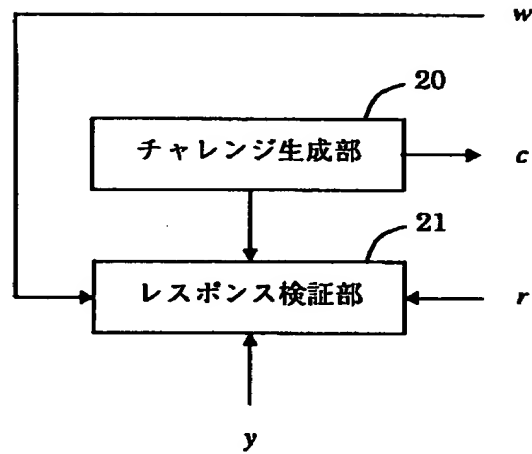
秘密鍵処理検証部

【図 10】



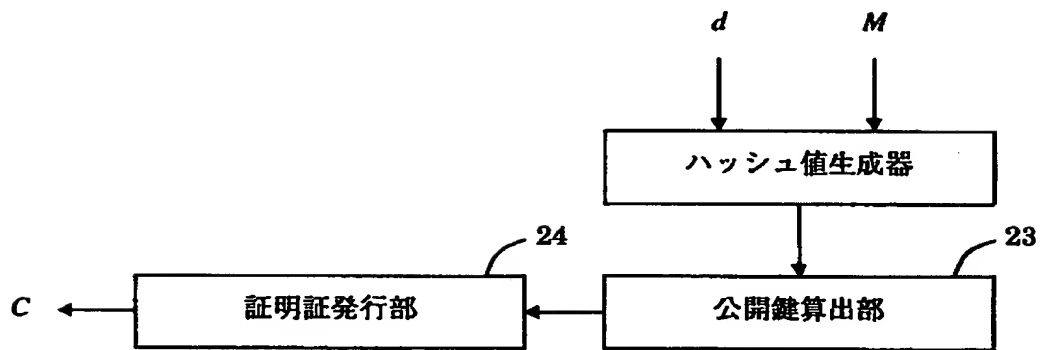
秘密鍵処理検証部

【図 1 1】



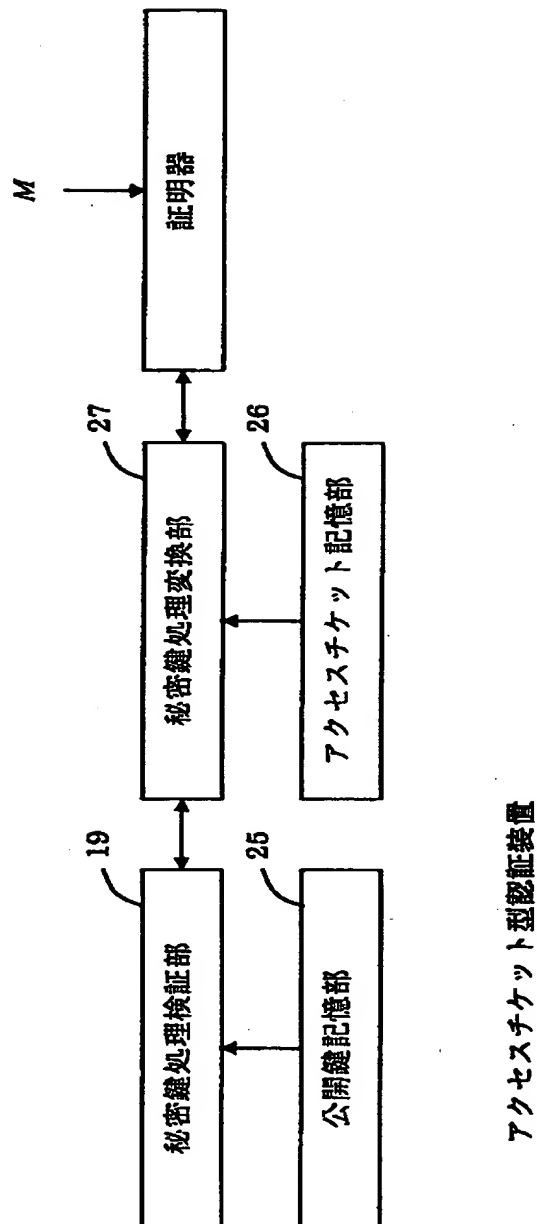
秘密鍵処理検証部

【図 1 2】

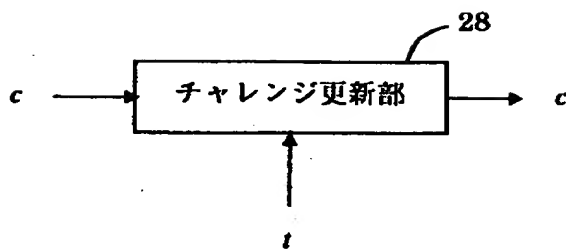


証明証発行装置

【図 1 3】

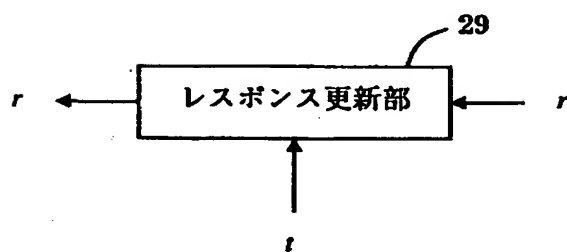


【図 1 4】



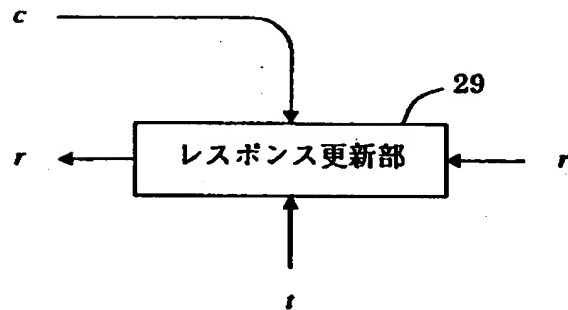
秘密鍵処理変換部

【図 1 5】



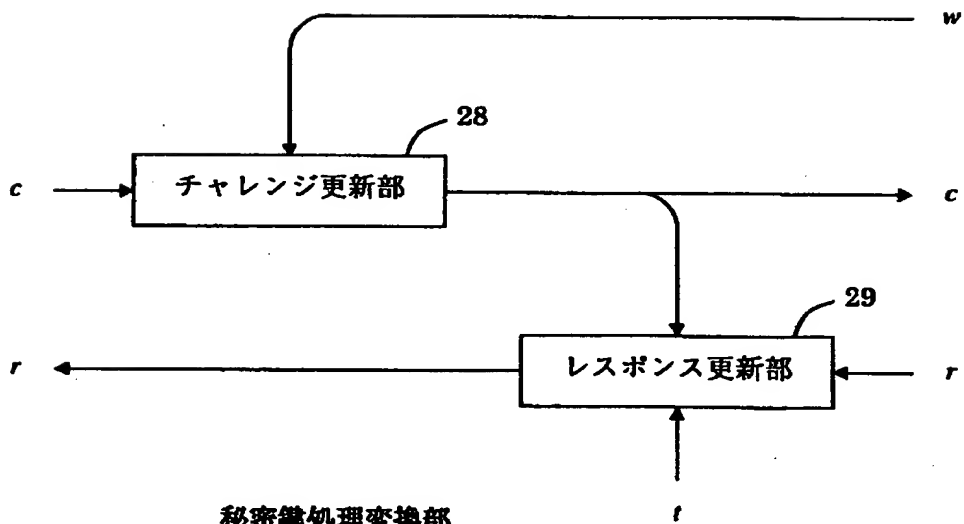
秘密鍵処理変換部

【図 1 6】



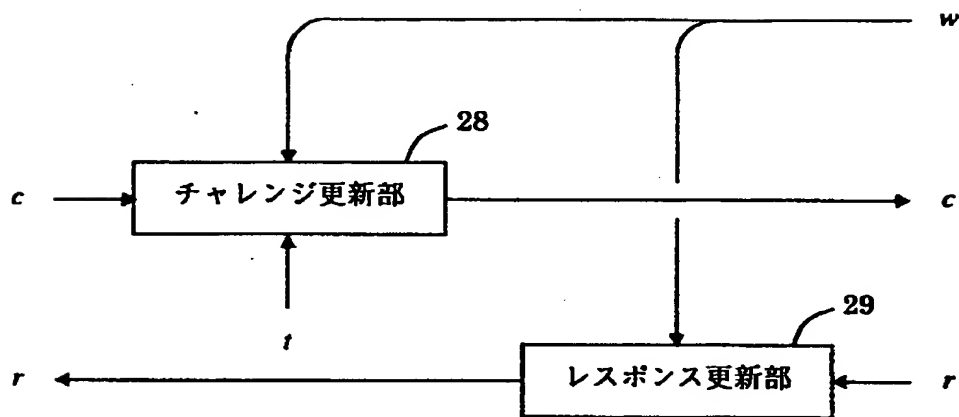
秘密鍵処理変換部

【図 1 7】



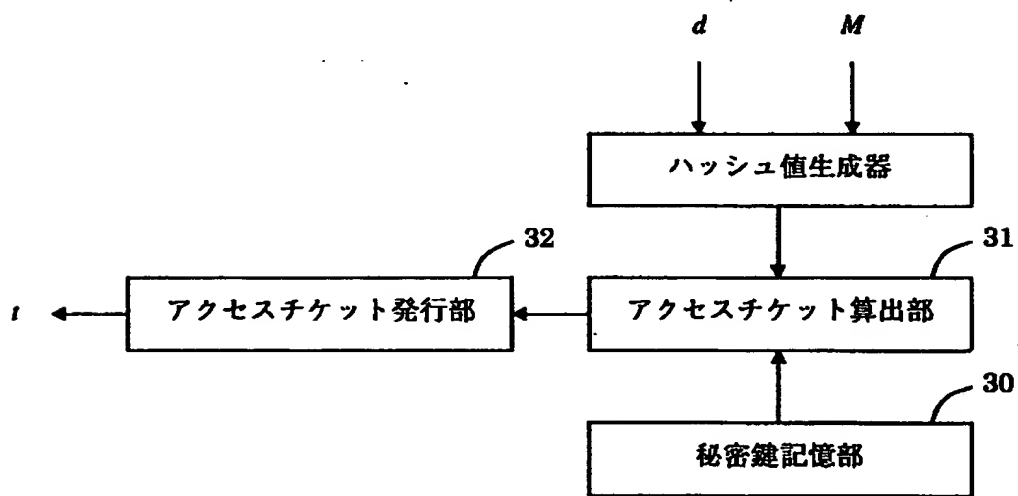
秘密鍵処理変換部

【図 1 8】



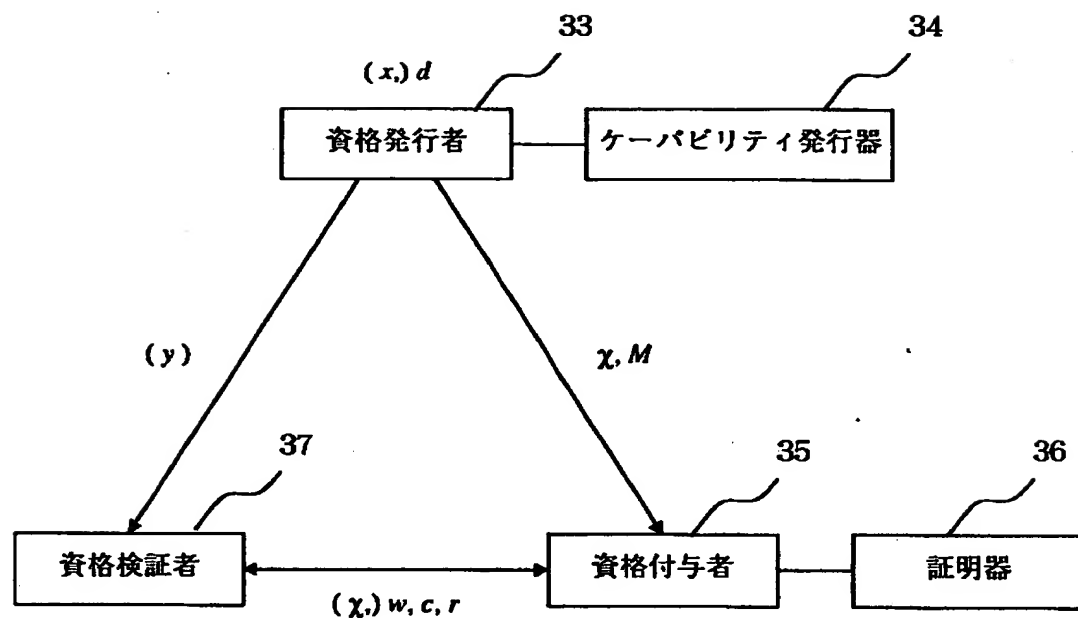
秘密鍵処理変換部

【図 1 9】



アクセスチケット発行装置

【図 2 0】



認証システム

【書類名】 要約書

【要約】

【課題】 トークンごとにハッシュ関数の秘密固有値を保管管理する必要なしに、かつセンターの組織的な秘密情報が暴露されるおそれなしに、トークンにハッシュ関数を付与する。

【解決手段】 固有値入力部 1 には、ハッシュ関数 X の生成に要するパラメータである固有値 d が入力される。メッセージ入力部 2 には、ハッシュ値を求めたいメッセージ M が入力される。関数生成固有値記憶部 3 は、値生成固有値の生成に要するパラメータである関数生成固有値 s を保持する。値生成固有値算出部 4 は、関数生成固有値 s と固有値 d より値生成固有値 u を生成する。ハッシュ値算出部 5 は、値生成固有値 u とメッセージ M よりハッシュ関数 H を用いてハッシュ値 X (M) を生成する。ハッシュ値出力部 6 は、ハッシュ値算出部 5 が生成したハッシュ値 X (M) を出力する。

【選択図】 図 1

特平11-244519

出 願 人 履 歴 情 報

識別番号 [000005496]

1. 変更年月日 1996年 5月29日

[変更理由] 住所変更

住 所 東京都港区赤坂二丁目17番22号

氏 名 富士ゼロックス株式会社